ＦＣＤＡ１６１６ＰＣＩ
取扱説明書

１６チャンネルバス絶縁方式

１２ビットＤ／Ａ変換基板

対応パソコン
IBM PC/AT互換機 （PCI）

合同会社　フレックスコア

２０１５年3月８日　ドライバーアンインストール操作方法追加
２０１４年5月１０日　WindowsXPインストール方法の見直し
２０１４年4月２９日　PCIデバイス指定方法修正
２０１４年1月１６日　ドライバーインストール方法を修正
２０１３年１２月２日発行　初版

フレックスコア

合同会社　フレックスコア

目　次

本製品の使用・適用についての注意事項 5
故障・修理・サポート方法について 6
本製品の構成 6
第１章　導入・試運転 7
１－１　本機の使用・概要 7
１－２　ボード上の設定 8
１－３　入出力コネクタ・ピン接続 8
アナログ入力 9
デジタル入出力 10
１－４　入出力接続オプション 10
１－５　ボードのインストール 10
１－６　試運転・動作確認 10
第２章　信号入出力 11
２－１　アナログ出力端 11
２－２　アナログ出力範囲 11
２－３　デジタル入出力 12
第３章　制御・操作 14
３－１　ＰＣＩデバイスコンフィギュレーション 14
３－２　制御・操作とアナログ出力の様子 14
３－３　制御レジスタＩ／Ｏアドレスマップ 15
３－４　ボードリセット（初期化）、認識 16
３－５　外部制御・割り込み関連の設定 16
３－６　ＤＡ出力モードの指定 17
３－７　アナログ出力範囲の設定 18
３－８　ＤＡ更新データの書き込み 19
３－９　ステータスデータ取得、クリア 20
３－10　汎用デジタル入出力 21
第４章　Windowsハンドラー 22
４－０　ハードウェアとソフトウェアの連携 22
４－１　本製品用ソフトウェアの種類 22
４－２　ドライバーのインストール 23
４－２－１　WindowsXPドライバーインストール及びアンインストール 23
４－２－２　Windows７へのインストール 26
４－３　Windows７ドライバーのアンインストール 29
~~４－４　動作確認プログラムについて（補足）~~ 31
４－５　関数仕様 32
【１】ボード、ハンドラーの初期化 32
【２】ボードのリセット 32
【３】DA出力モードの設定 32
【４】DA出力範囲、コードの指定 33
【５】DA即時更新出力 33
【６】ボードステータスの取得 33
【７】本ハンドラーの終了 33
【８】汎用デジタル出力 34
【９】汎用デジタル入力 34

【10】デジタル入力による強制０Ｖ出力.....34
【11】全チャンネル同期更新命令.....34
【12】バージョン情報取得.....34
エラーコード一覧.....35
第５章　汎用デバイスドライバー仕様書.....36
【１】ドライバーオープン.....37
【２】ドライバースタート.....37
【３】バイトリード.....37
【４】バイトブロックリード.....37
【５】ワードリード.....38
【６】ワードブロックリード.....38
【７】ダブルワードリード.....38
【８】ダブルワードブロックリード.....38
【９】バイトライト.....39
【10】バイトブロックライト.....39
【11】ワードライト.....39
【12】ワードブロックライト.....39
【13】ダブルワードライト.....40
【14】ダブルワードブロックライト.....40
【15】ドライバーストップ.....40
【16】ドライバークローズ.....40
エラーコード.....41
第６章　保守、その他.....41
６－１　故障、トラブル等の原因と対処.....41
６－２　修理のときは.....41
６－３　アナログ出力範囲の再調整.....43
６－３－１　機器間の接続.....43
６－３－２　暖機運転.....43
６－３－３　調整作業の実際.....43
第７章　エラー情報.....44
７－１　PCIデバイス指定方法について.....44

本製品の使用・適用についての注意事項

（1） 本製品は、ＩＢＭＰＣ／ＡＴ互換機のＰＣＩバス拡張Ｉ／Ｏスロット、またはＰＣＩバス拡張Ｉ／Ｏボックスに装着して使用するものです。

（2） 本製品が組み込まれたシステムの運用対象・方法・場所・環境等によって、故障・誤動作等が生じた場合に起こり得る、身体・生命・財産等に対する損害の回避処置は同システムの設計・製作に別途付加・反映させて下さい。本製品自体には、前述の機能はなく、従って弊社では本製品が組み込まれたシステムの運用により発生した故障・誤動作・事故に起因する身体・生命・財産等の損害に対する責任は負えません。これは本製品の故障・誤動作が原因となった場合も含み、理由の如何を問いません。

（3） 本製品付属のソフトウェアは本製品利用の方法を示す例、またオプションの関連ソフトウェアは本製品利用の一般的便宜をはかるものであり、現在未発見のバグ存在の可能性も含めて、運用結果についての責任は一切負えません。
これらのソフトウェアには自身が組み込まれたシステムに故障・誤動作・事故等が生じた場合に起こり得る身体・生命・財産等に対する損害の回避機能はありません。御利用の場合は同システムの設計・製作で配慮・付加・反映させて下さい。

（4） 本製品（付属ソフトウェアを含む）、及びオプションの関連ソフトウェアは医用・航空機器用・その他高信頼性・高安全性を必要とするシステムに使用しないで下さい。

（5） 本製品付属のソフトウェアについて弊社は著作権を保持しますが、第３者の権利を侵害しない限りにおいて購入者は自身が製作するシステム等に自由に組み込み、販売する事もできます。但し、弊社製ソフトウェアのソースコードを含むソフトウェアを第３者に販売・移転するときは弊社の文書による事前許可を必要とします。

（6） 弊社では、本製品の販売・サポート・保証の範囲を日本国内に限っています。

（7） 本製品は改良のため仕様変更、価格改定を行うことがあります。

故障・修理・サポート方法について

（1）　納入後１年間は自然故障、及び弊社製造上の問題に起因した事が明らかな故障製品に対して無償修理を行います。

（2）　保証期間中であっても、次の場合は有償修理となります。

ア　取扱上の不注意、誤用による故障および損傷
イ　弊社以外での修理、改造、分解掃除等による故障および損傷
ウ　泥、砂、水などのかぶり、落下、衝撃等が原因で発生した故障および損傷
エ　火災、地震、水害、落雷その他の天変地異、公害や異常電圧による故障および損傷
オ　保管上の不備（高温多湿の場所等）や手入れの不備による故障
カ　接続している他の機器に起因して生じた故障
キ　その他使用者側の責に帰する故障

（3）　修理は宅配便によるセンドバックで行います。尚運賃は互いに発送する側が負担するものとします。
　出張修理は行っておりません。簡単な故障であれば一週間程度で修理・返却が可能ですが、故障状況によっては更に日数を要します。

（4）　本製品使用上の質問・トラブル対応・故障修理などは入手経路の如何にかかわらず、弊社宛に直接御相談下さい。その際は、客観情報の整理・評価を行うために必ずレポートを御送付下さい。特にEmailで情報を頂く事で解決速度が大幅に向上する事が期待できます。（support@flexcore.jp）また、内容が複雑になっている場合はfax使用も有効です。

（5）　有償修理の場合の修理費用は、基本料金￥１０，０００円＋部品代となります。簡単な故障では凡そ２万円以下程度だと思われます。修理費用限度額がある場合は、お申し付け頂ければ超過する場合に御連絡致します。

（6）　修理品送付先

〒301-0853
茨城県龍ヶ崎市松ヶ丘３丁目１８番地３
フレックスコア
品質管理部　林
E-mail: support@flexcore.jp
Fax:050-3488-3354

本製品の構成

本製品は　ＦＣＤＡ１６１６ＰＣＩボード
入出力プラグ（アナログ、デジタル各１個）
からなります。
ドライバーソフトウェア、取扱説明書（本文書）、サンプルプログラムなどについては、
弊社ホームページよりダウンロードをお願いします。

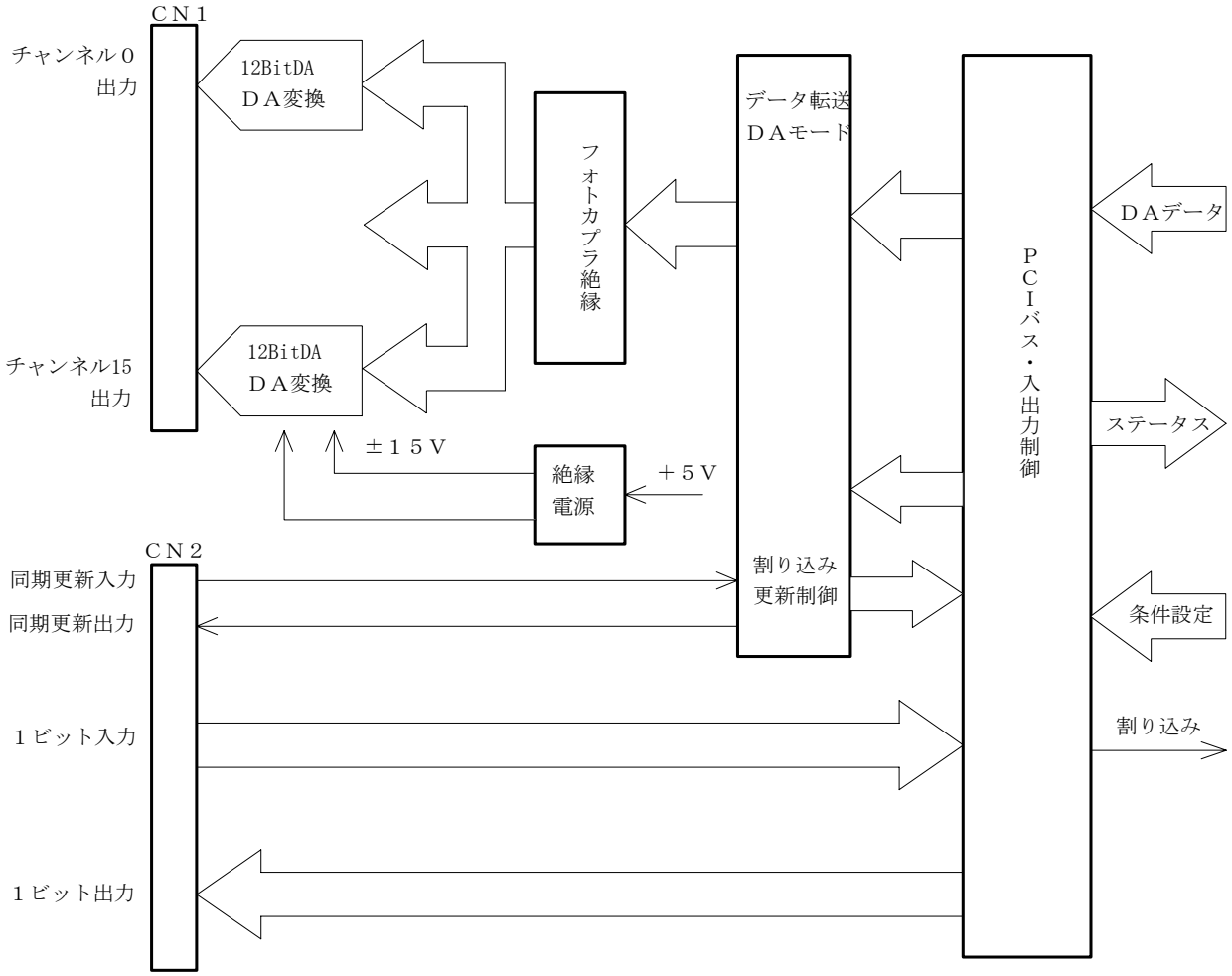
第１章　導入・試運転

１－１　本機の使用・概要

本ボードは、ＩＢＭ／ＡＴパソコンに装着され、最大１６枚の同期運転を行う全チャンネル共通バス絶縁型１２ビットＤＡボードです。ボード単体としては、４チャンネル単位で最大１６チャンネルまでのＤＡ出力を型式指定する事ができます。各ＤＡ出力は単独、または同期して更新出力を行う事ができ、更に複数の本ボード間でも同期更新する事ができます。

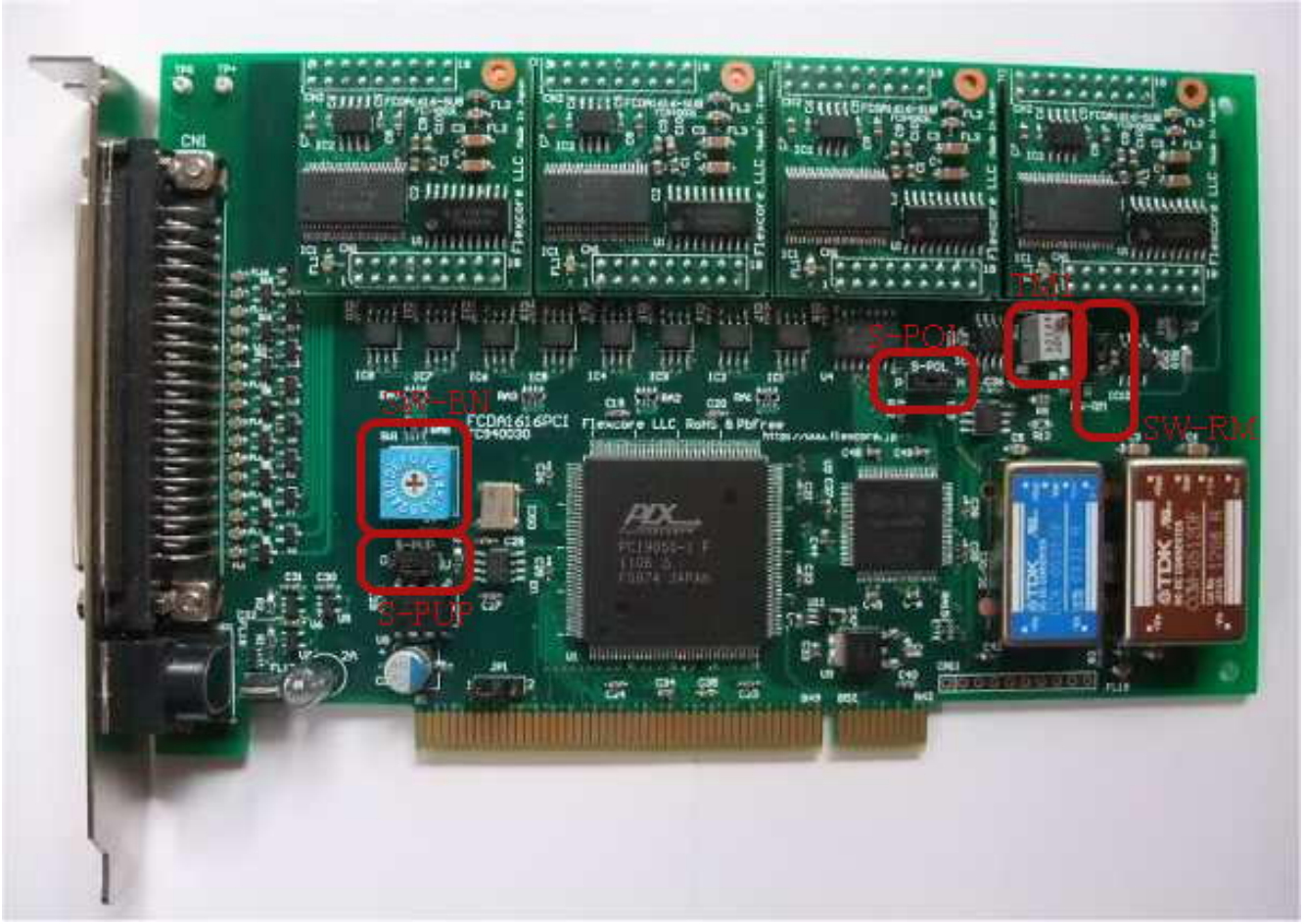
分解能：１２ビット
セトリング時間：１０ｕＳ（０．１％ＦＳ／負荷容量１００ｐＦで１０Ｖスイングの時）
非直線性：０．００４％ＦＳ
正確度１：０．０７％ＦＳ（Ａモード）
正確度２：０．０９％ＦＳ（Ｂモード）
温度ドリフト：０．００２％ＦＳ／℃
電圧出力時の負荷駆動能力：容量５００ｐＦ以下、抵抗５ＫΩ以上（負荷電流２ｍＡ以下）
出力電圧範囲：±１０Ｖ／±５Ｖ／０～＋１０Ｖ／０～＋５Ｖ（ソフト設定）
ＤＡ出力動作：全チャンネル同期／各チャンネル単独
グリッチ・インパルス：最大１２０ｍＶ（±１０Ｖ出力範囲の時）

図１－１　ＦＣＤＡ１６１６ＰＣＩ機能ブロック



１－２　ボード上の設定

本ボード上の設定は、アナログ出力範囲選択ジャンパー（ＳＷ－ＲＭ）、デジタル出力の論理切り替えジャンパー（Ｓ－ＰＯＬ）、デジタル出力プルアップ切り替えジャンパー（Ｓ－ＰＵＰ）、アナログゲイン調整トリマ（ＴＭ１）及びボード番号設定スイッチ（ＳＷ－ＢＮ）からなっています。



ＳＷ－ＢＮ：ボード番号設定スイッチ（出荷時：０）
Ｓ－ＰＵＰ：汎用デジタル出力のプルアップ有無（出荷時：Ｕ（プルアップ接続））
Ｓ－ＰＯＬ：汎用デジタル出力の論理極性選択（出荷時：Ｐ（正論理））
ＳＷ－ＲＭ：アナログ出力範囲選択ジャンパー（出荷時：Ａ側）
ＴＭ１：アナログゲイン調整トリマ
ＦＳ１：＋５Ｖ電源出力保護用ヒューズ（２Ａ：浜井電球製）

１－３　入出力コネクタ・ピン接続

アナログ入力には３７ピン D-SUB コネクタ、また汎用デジタル入出力を含む制御信号入出力には８ピン丸型コネクタを使用しており適合プラグ各１個が添付されています。

アナログ入力

基板側：１７ＬＥ－１３３７０－２７（Ｄ４ＡＢ）　ＤＤＫ製・実装

図１－３Ａ　アナログ入力コネクタ（ＣＮ１）ピン接続

信号名（機能）	ピン番号		ピン番号	信号名（機能）
ＣＨ０出力	１	○		
			○ ２０	ＡＧ（アナロググランド）
ＣＨ１出力	２	○		
			○ ２１	ＡＧ（アナロググランド）
ＣＨ２出力	３	○		
			○ ２２	ＡＧ（アナロググランド）
ＣＨ３出力	４	○		
			○ ２３	ＡＧ（アナロググランド）
ＣＨ４出力	５	○		
			○ ２４	ＡＧ（アナロググランド）
ＣＨ５出力	６	○		
			○ ２５	ＡＧ（アナロググランド）
ＣＨ６出力	７	○		
			○ ２６	ＡＧ（アナロググランド）
ＣＨ７出力	８	○		
			○ ２７	ＡＧ（アナロググランド）
ＣＨ８出力	９	○		
			○ ２８	ＡＧ（アナロググランド）
ＣＨ９出力	１０	○		
			○ ２９	ＡＧ（アナロググランド）
CH10 出力	１１	○		
			○ ３０	ＡＧ（アナロググランド）
CH11 出力	１２	○		
			○ ３１	ＡＧ（アナロググランド）
CH12 出力	１３	○		
			○ ３２	ＡＧ（アナロググランド）
CH13 出力	１４	○		
			○ ３３	ＡＧ（アナロググランド）
CH14 出力	１５	○		
			○ ３４	ＡＧ（アナロググランド）
CH15 出力	１６	○		
			○ ３５	ＡＧ（アナロググランド）
空きピン	１７	○		
			○ ３６	空きピン
空きピン	１８	○		
			○ ３７	ＤＧ（デジタルグランド）
＋５Ｖ出力	１９	○		

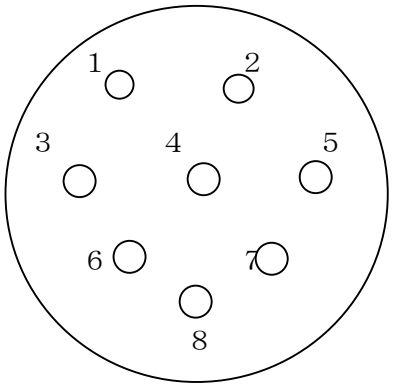
注　アナロググランドとデジタルグランドはボード内部で接続されています。

プラグ：１７ＪＥ－２３３７０－０２（Ｄ８Ａ）－ＣＧ

デジタル入出力

コネクタ　基板側：ＨＲ２１２－１０ＲＡ－８ＳＤＬ（７４）　ヒロセ製

図１－３Ｃ　デジタル入出力コネクタ・ピン接続



添付プラグの半田付け側面視

１ピン：
２ピン：
３ピン：同期制御入力
４ピン：同期制御出力
５ピン：汎用デジタル入力
６ピン：汎用デジタル出力
７ピン：＋５Ｖ出力
８ピン：デジタルグランド

１－４　入出力接続オプション

本ボードにはアナログ出力用及びデジタル入出力用にプラグが各１個添付されていますが、アナログ接続オプションとして 16 チャンネルＢＮＣ接続ボックス（ＣＢＯＸ－０１４　１６ＣＨ）が用意されています。

１－５　ボードのインストール

本製品はプラグアンドプレイに対応した PCI ボードです。そのため御使用に先立ち組み込むパソコンにインストール（認識、リソース割当て）する必要があります。そのためのステップは次の通りです。

１　まず、弊社ホームページよりドライバーパッケージをダウンロードします。そして、このパッケージを展開解凍し、その後インストール作業を行います。この際、本ボードはまだパソコンに取り付けてはいけません。

２　デバイスドライバーをインストールした後、一旦パソコンの電源を落とし、その後本ボードを PCI スロットに取り付けてからパソコンの電源を再投入します。一旦システムトレイにドライバーインストール中のメッセージが表示され、一分ほど待っていると、ドライバーがインストールされ、ハードウェアの使用準備ができた旨、表示されます。

３　この段階で本ボードを使用する事ができる状態になっていますので、必要に応じ適宜弊社ホームページよりサンプルプログラムをダウンロードして頂き実行モジュールを作成するか、或いはご自身で使用される言語システムを利用した開発を開始する事ができます。より詳細な内容については４－１章を参照して下さい。

１－６　試運転・動作確認

Windows7 対応の動作確認プログラムを用意しています。詳細は弊社ホームページよりダウンロードできるソフトウェアキットを参照してください。

第２章　信号入出力

２－１　アナログ出力端

本機のアナログ出力回路は全チャンネル共通の形でアイソレーターを使用しパソコン側から絶縁されています。アナログ回路側の電源（±１５Ｖ及び＋５Ｖ）は、パソコン内部のロジック電源（＋５Ｖ）から絶縁型ＤＣＤＣコンバーターによって供給されています。更にアナログ出力端には EMI フィルタ及び高速ダイオードによるノイズ対策回路が付加されています。

２－２　アナログ出力範囲

アナログ入力範囲は、表２－２Ａに示す範囲の中から選択する事ができます。１２ビット分解能は、１／４０９６ですが、公称入力範囲で正直に調整するとＤＡ変換値１デジット当たりの電圧値が割り切れない値となります。そこで、Ａモードと称し、公称入力範囲を４０００で除した分解能を採用したモードも用意しています。

表２－２Ａ　１２ビットモードでの入力範囲・分解能・正確度

公称入力範囲	範囲モード	実際の入力範囲 （±１０Ｖを越えた部分は無効）	分解能 mV/digit	正確度
±１０Ｖ	Ａ	－１０．２４Ｖ～＋１０．２３５Ｖ	５	表２－２Ｃ１
	Ｂ	－１０．０００Ｖ～＋９．９９５１２Ｖ	４．８８	
±５Ｖ	Ａ	－５．１２Ｖ～＋５．１１７５Ｖ	２．５	
	Ｂ	－５．０００Ｖ～＋４．９９７５６Ｖ	２．４４	
０～＋１０Ｖ	Ａ	０Ｖ～＋１０．２３７５Ｖ	２．５	
	Ｂ	０Ｖ～＋９．９９７５６Ｖ	２．４４	
０～＋５Ｖ	Ａ	０Ｖ～＋５．１１８７５Ｖ	１．２５	
	Ｂ	０Ｖ～＋４．９９８７８Ｖ	１．２２	

表２－２Ｂ　１２ビットＤＡデータｖｓアナログ出力（Ａモード）

ＡＤデータ	アナログ入力範囲					
Hex/10 進	±１０Ｖ	±５Ｖ	±２．５Ｖ	０～＋１０Ｖ	０～＋５Ｖ	０～＋２．５Ｖ
FFF/4095	+10.235	+5.1175	+2.55875	+10.2375	+5.11875	+2.55938
FD0/4048	+10.000	+5.0000	+2.5000			
FA0/4000				+10.0000	+5.0000	+2.50000
801/2049	+0.005	+0.0025	+0.00125			
800/2048	0.000	0.0000	0.00000			
7FF/2047	-0.005	-0.0025	-0.00125			
7D0/2000				+5.0000	+2.50000	+1.25000
030/48	-10.000	-5.0000	-2.50000			
001/1	-10.235	-5.1175	-2.55875	+0.0025	+0.00125	+0.000625
000/0	-10.24	-5.1200	-2.56000	0.0000	0.00000	0.00000

表２－２Ｃ　１２ビットＤＡデータｖｓアナログ出力（Ｂモード）

ＡＤデータ	アナログ入力範囲					
Hex/10 進	±１０Ｖ	±５Ｖ	±２．５Ｖ	０～＋１０Ｖ	０～＋５Ｖ	０～＋２．５Ｖ
FFF/4095	+9.99512	+4.99756	+2.49878	+9.99756	+4.99878	+2.49939
FD0/4048	+9.76563	+4.88281	+2.44141			
FA0/4000				+9.76563	+4.88281	+2.44141
801/2049	+0.00488	+0.00244	+0.00122			
800/2048	0.00000	0.0000	0.00000	+5.00000	+2.50000	+1.25000
7FF/2047	-0.00488	-0.00244	-0.00122			
7D0/2000				+4.88281	+2.44141	+1.22070
030/48	-9.76563	-4.88281	-2.44141			
001/1	-9.99512	-4.99756	-2.49878	+0.00244	+0.00122	+0.00061
000/0	-10.00000	-5.00000	-2.50000	0.00000	0.00000	0.00000

伝達関数

１２ビットＤＡの分解能は１／2^{12}ですから、ＤＡデータとアナログ出力電圧の関係は以下の様になります。

分解能　　$Res = V_{span} \div 4096$　　(V/Digit)

ＤＡデータ　$D_{da} = V_{out} \div Res$　　(digit)　　ユニポーラの時

$D_{da} = (V_{out} \div Res) + 2048$　　バイポーラの時

出力電圧　　$V_{out} = D_{da} \times Res$　　(V) ユニポーラの時

$V_{out} = (D_{da} - 2048) \times Res$　(V) バイポーラの時

２－３　デジタル入出力

同期更新入力、外部割込み入力、汎用デジタル入力は全てＴＴＬレベルであり、１０ＫΩでプルアップされています。同期更新出力はＴＴＬレベル、汎用の１ビットデジタル出力はオープンコレクタですが、接続先の事情によってはボード上のジャンパーＳ－ＰＵＰで１ＫΩのプルアップ抵抗を接続する設定（出荷時状態）も可能です。

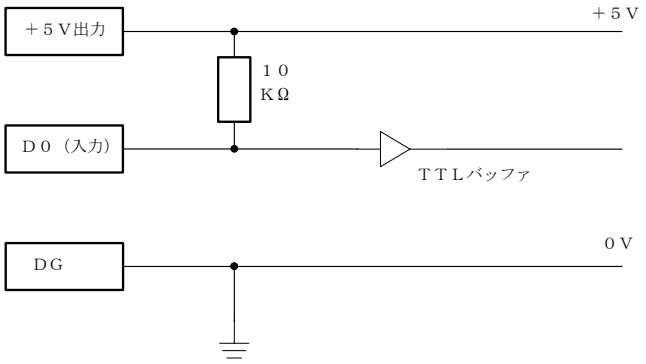


図２－３Ａ　デジタル入力模式図

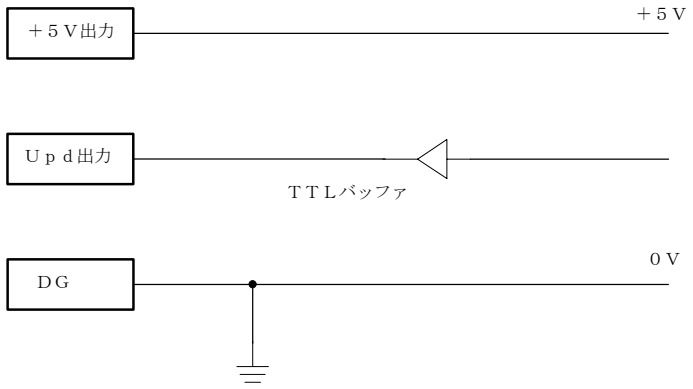


図２－３Ｂ　同期更新出力

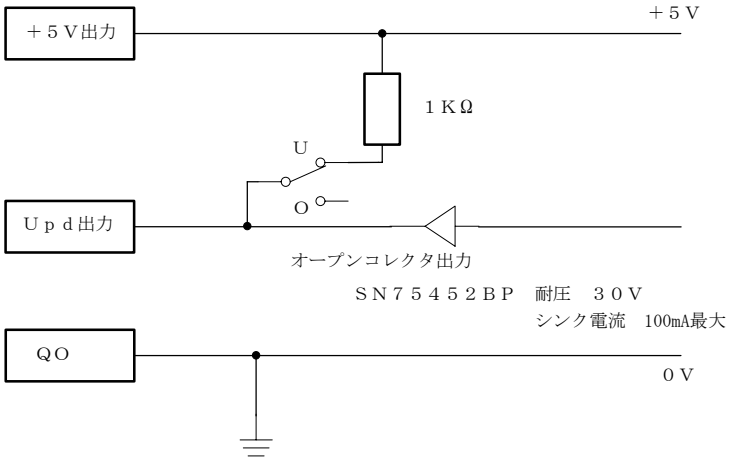


図２－３Ｃ　汎用デジタル出力

第３章　制御・操作

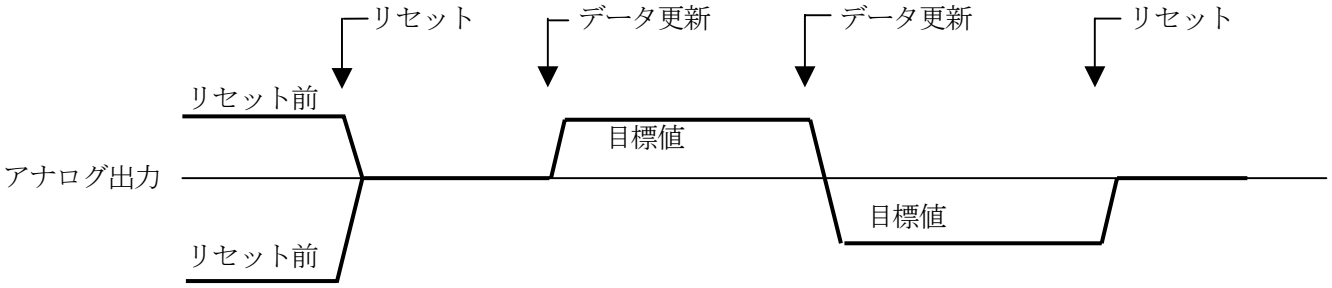
３－１　ＰＣＩデバイスコンフィギュレーション

本ボードのPCIデバイスコンフィギュレーション情報は、ベンダーID：１３ＦＤ、デバイスID：０１０Ｅとなっています。基本的にこの情報はドライバー内部に組み込まれているものですが、ユーザーの便宜を図り記述しています。

３－２　制御・操作とアナログ出力の様子

電源投入、パソコン本体のリセット、或いは本ボード制御部のリセット動作直後のアナログ出力電圧は０Ｖとなります。更新出力後（任意のデータがＤＡ素子に書き込まれその出力が更新される）とアナログ出力は対応する値となり、以後次の更新までその値を保持します。

図３－１Ａ　アナログ出力の変化



データ更新遅れ：パソコン側からＯＵＴ命令で書き込まれるＤＡデータはアイソレーター（転送時間≒１ｕＳ）を通じてＤＡ素子のラッチに伝達されます。

セトリング時間:データが更新されると、ＤＡ素子のアナログ出力は新たな目標値に向かって変化を開始します。本機では、１０Ｖの変化幅を０．１％ＦＳ精度で到達する時間（セトリング時間）が約８ｕＳですが、上に述べたアイソレーターの転送時間を加え余裕を見て１０ｕＳとしています。

単独更新　　：各ＤＡチャンネルが非同期に独立してその出力を更新制御する動作モードです。

同期更新　　：各チャンネルにＤＡ出力データを書き込み、最後に同期更新操作を行うことで全チャンネルのＤＡ出力が一斉に更新される動作モードです。

複数ボードの同期更新

ハードウェア上の設定：マスターボードの同期更新出力をスレーブボードの同期更新入力に接続します。

ソフトウェア上の設定：スレーブボードの更新モード選択を同期に、更新実行要素を外部更新入力にセットしマスターボードの更新モードも同期に、更新実行要素をソフト操作にセットします。

ＤＡ出力操作　　：スレーブ、マスター夫々のＤＡ出力データ書込み操作を全て行い、最後にマスターボードのソフトウェア同期更新操作を行います。その結果全てのＤＡ出力が新しい値に更新動作を開始します。

３－３　制御レジスタＩ／Ｏアドレスマップ

表３－２に本ボード上の各制御レジスタのＩ／Ｏアドレスを示します。

表３－２Ａ制御出力レジスタＩ／Ｏアドレス

Ｉ／Ｏアドレス	出力ポート・レジスタ名	情報掲載項目等
BASE＋２Ｅ	ＤＡチャンネル１５／出力データ	３－７項
BASE＋２Ｃ	〃　　１４／〃	
BASE＋２Ａ	〃　　１３／〃	
BASE＋２８	〃　　１２／〃	
BASE＋２６	〃　　１１／〃	
BASE＋２４	〃　　１０／〃	
BASE＋２２	〃　　９／〃	
BASE＋２０	〃　　８／〃	
BASE＋１Ｅ	〃　　７／〃	
BASE＋１Ｃ	〃　　６／〃	
BASE＋１Ａ	〃　　５／〃	
BASE＋１８	〃　　４／〃	
BASE＋１６	〃　　３／〃	
BASE＋１４	〃　　２／〃	
BASE＋１２	〃　　１／〃	
BASE＋１０	〃　　０／〃	
BASE＋０Ｅ	マニュアル同期更新制御	３－７項
BASE＋０Ｃ	汎用１ビット出力	３－９項
BASE＋０Ａ		
BASE＋０８	ステータス（割り込み要求認識フラグ）クリア	３－８項
BASE＋０６	ＤＡチャンネル８～１５／出力範囲選択	３－６項
BASE＋０４	ＤＡチャンネル０～７／出力範囲選択	３－６項
BASE＋０２	外部割込み入力、割り込み制御・設定	３－４項
BASE＋００	ＤＡ出力モード設定、ＤＡデータコード選択	３－５項

表３－２Ｂ　制御入力レジスタＩ／Ｏアドレス

Ｉ／Ｏアドレス	入力ポート・レジスタ名	情報掲載項目等
BASE＋　Ｅ	ボード制御部リセット	３－３項
BASE＋　Ｃ	汎用１ビット入力	３－９項
BASE＋　Ａ		
BASE＋　８	ステータス取得	３－８項
BASE＋　６		
BASE＋　４	ボード番号（ＳＷ－ＢＮの設定値）取得	３－３項
BASE＋　２	（ＰＣＩバスへの）割り込み出力信号モニタ	３－４項
BASE＋　０		

３－４　ボードリセット（初期化）、認識

ｒｓｔ＝ｉｎｐ（ＢＡＳＥ＋０ｘＥ）；

本ボードの制御部全体を初期化します。この操作で読み込まれるデータは、このボードのボードＩＤです。この操作は汎用１ビットデジタル出力の状態が保持される事を除き、電源投入またはパソコン本体のリセット操作と同等の機能です。

主な機能は

・各制御レジスタの初期化
・ボードステータスの初期化
・アナログ出力は全チャンネル０Ｖとなる

となります。

表３－３　【ＢＡＳＥ＋０ｘＥ】入力ポートの構成

ビット	各ビットの機能・意味
Ｂ１５～Ｂ８	未使用
Ｂ７～Ｂ０	ボードＩＤ（１７Ｈ）

注　上記ボードＩＤはＰＣＩバス上のデバイスＩＤとは無関係です。

本ボードを複数使用する場合

本ボードが使用するアドレス（主としてＩＯアドレス）は、プラグアンドプレイ機能により毎回ダイナミックに割り当てられます。そこでこのことによってボード相互の関係が変化する事の無いよう、ボード番号設定スイッチが用意されました。このスイッチの値を予めボード毎に異なる値に設定しておく事で、アプリケーション側から見ると常に同一のボード相互関係が保証される事になります。

ＢＮ＝ｉｎｐ（ＢＡＳＥ＋０Ｘ４）

表３－５Ｂ【ＢＡＳＥ＋０ｘ４】入力ポートの設定

ビット	各ビットの機能・意味
Ｂ１５～Ｂ４	未使用
Ｂ３～Ｂ０	ボード番号設定スイッチの値（０～ＦＨ）

３－５　外部制御・割り込み関連の設定

ｏｕｔｐ（ＢＡＳＥ＋０ｘ２、ＣＮＴ）

本機のアナログ（ＤＡ）出力更新を外部入力信号（UPD_IN）で行う場合、及び同信号を汎用の割り込みに利用する時の許可・禁止・信号極性及び割り込み要求信号出力制御（許可／禁止）を行います。

何れの機能も使用しない場合、本ポートの操作を行う必要はありません。

表３－４A【ＢＡＳＥ＋０ｘ２】出力ポートの構成

ビット	各ビットの機能・意味	＝１の時	＝０の時	リセット時
B15～B8	未使用	（		
Ｂ７	外部入力制御（割り込み入力）	許可	禁止	０
Ｂ６	外部入力信号の有効エッジ指定	立上り（↑）	立下り（↓）	０
Ｂ５	汎用デジタル入力による強制０Ｖ制御	許可	禁止	０
B4～B1	未使用			
Ｂ０	ＰＣＩバス上のＩＮＴ信号出力制御	許可	禁止（＝クリア）	０

補足説明

Ｂ７：外部入力信号【UPD_IN】を汎用割り込み要求として使用する時にはこのビットをセットする必要があります。

Ｂ６：外部入力信号【UPD_IN】の有効極性を指定するビットです。汎用の割り込み入力として許可されている場合はこのビットで指定された極性でボードステータスの外部割込み入力認識フラグがセットされます。

Ｂ５：このビットをセットする事により、汎用デジタル入力の状態が開放或いはＴＴＬハイレベルとなった時点で全ＤＡ出力がリセット状態（出力＝０Ｖ）になります。

Ｂ０：実際のＰＣＩバス上の割り込み要求信号を許可するかどうかを制御するレジスタです。このビットが０の状態になっているとＰＣＩバスへの割り込み要求出力は禁止されます。

割り込み信号モニター

ｉｎｔ＝ｉｎｐ（ＢＡＳＥ＋０ｘ２）；　　// 割り込み信号出力ステータス

本ボードからＰＣＩバスへ出力されている割り込み要求信号の状態をモニターするポートです。

表３－４Ｂ【ＢＡＳＥ＋０ｘ２】入力ポートの構成

ビット	各ビットの機能・意味	＝１の時	＝０の時	リセット時
B15~B1	未使用	0		0
B0	ＰＣＩバス上の割り込み信号ステータス	アクティブ	非アクティブ	0

３－６　ＤＡ出力モードの指定

ｏｕｔｐ（ＢＡＳＥ＋０ｘ０H、MD）

ＤＡ出力データ更新モード、ＤＡデータコードの設定を行います。（全チャンネル共通）

表３－５【ＢＡＳＥ＋０】出力ポートの構成

ビット	各ビットの機能・意味	＝１の時	＝０の時	リセット時
B15~B8	未使用			
B7	ＤＡ出力データ更新モード	同期更新	単独更新	0
B6	ＤＡ出力データ同時更新実行要素	外部更新入力	ソフト操作	0
B5	ＤＡデータ・コード（バイポーラの時）	２の補数	オフセットバイナリ	0
B4～B0	未使用			

補足説明

Ｂ７：本ビットをセットすると、全チャンネルのＤＡ出力更新はビット６で指定する要素により同時に行われます。

Ｂ６：本ビットがセットされていると、外部入力（ＵＰＤ＿ＩＮ）によって、またリセットされていると別項のソフト操作により全チャンネルが同時に更新されます。

Ｂ５：ユニポーラ出力範囲のＤＡデータコードはストレートバイナリのみですが、バイポーラ出力の時はこのビットによりデータコードの選択ができます。

３－７　アナログ出力範囲の設定

ｏｕｔｐ（ＢＡＳＥ＋４、ＲＡＬ）
ｏｕｔｐ（ＢＡＳＥ＋６、ＲＡＨ）

各ＤＡ出力チャンネルの範囲を選択・指定します。

表３－６Ａ　各ＤＡチャンネルアナログ出力範囲指定ポートの構成

ビット	コード	【ＢＡＳＥ＋４Ｈ】出力ポート	【ＢＡＳＥ＋６Ｈ】出力ポート	リセット時
B15	R1	チャンネル７出力範囲設定	チャンネル 15 出力範囲設定	0
B14	R0			0
B13	R1	チャンネル６出力範囲設定	チャンネル 14 出力範囲設定	0
B12	R0			0
B11	R1	チャンネル５出力範囲設定	チャンネル 13 出力範囲設定	0
B10	R0			0
B9	R1	チャンネル４出力範囲設定	チャンネル 12 出力範囲設定	0
B8	R0			0
B7	R1	チャンネル３出力範囲設定	チャンネル 11 出力範囲設定	0
B6	R0			0
B5	R1	チャンネル２出力範囲設定	チャンネル 10 出力範囲設定	0
B4	R0			0
B3	R1	チャンネル１出力範囲設定	チャンネル９出力範囲設定	0
B2	R0			0
B1	R1	チャンネル０出力範囲設定	チャンネル８出力範囲設定	0
B0	R0			0

表３－６Ｂ　出力範囲指定コード

R1	R0	アナログ出力範囲
1	1	±５Ｖ
1	0	±１０Ｖ
0	1	０～＋５Ｖ
0	0	０～＋１０Ｖ

注　アナログ出力範囲には（ボード上のジャンパー設定により）
　　２種類のモードがあり夫々出力範囲に若干の違いがあります。

３－８　ＤＡ更新データの書き込み

ｏｕｔｐ（ＢＡＳＥ＋０ｘ１０、ＤＡＤ０）// ＤＡチャンネル０データの書き込み
ｏｕｔｐ（ＢＡＳＥ＋０ｘ１２、ＤＡＤ１）// ＤＡチャンネル１データの書き込み
ｏｕｔｐ（ＢＡＳＥ＋０ｘ１４、ＤＡＤ２）// ＤＡチャンネル２データの書き込み
ｏｕｔｐ（ＢＡＳＥ＋０ｘ１６、ＤＡＤ３）// ＤＡチャンネル３データの書き込み
ｏｕｔｐ（ＢＡＳＥ＋０ｘ１８、ＤＡＤ４）// ＤＡチャンネル４データの書き込み
ｏｕｔｐ（ＢＡＳＥ＋０ｘ１Ａ、ＤＡＤ５）// ＤＡチャンネル５データの書き込み
ｏｕｔｐ（ＢＡＳＥ＋０ｘ１Ｃ、ＤＡＤ６）// ＤＡチャンネル６データの書き込み
ｏｕｔｐ（ＢＡＳＥ＋０ｘ１Ｅ、ＤＡＤ７）// ＤＡチャンネル７データの書き込み
ｏｕｔｐ（ＢＡＳＥ＋０ｘ２０、ＤＡＤ８）// ＤＡチャンネル８データの書き込み
ｏｕｔｐ（ＢＡＳＥ＋０ｘ２２、ＤＡＤ９）// ＤＡチャンネル９データの書き込み
ｏｕｔｐ（ＢＡＳＥ＋０ｘ２４、ＤＡＤ10）// ＤＡチャンネル 10 データの書き込み
ｏｕｔｐ（ＢＡＳＥ＋０ｘ２６、ＤＡＤ11）// ＤＡチャンネル 11 データの書き込み
ｏｕｔｐ（ＢＡＳＥ＋０ｘ２８、ＤＡＤ12）// ＤＡチャンネル 12 データの書き込み
ｏｕｔｐ（ＢＡＳＥ＋０ｘ２Ａ、ＤＡＤ13）// ＤＡチャンネル 13 データの書き込み
ｏｕｔｐ（ＢＡＳＥ＋０ｘ２Ｃ、ＤＡＤ14）// ＤＡチャンネル 14 データの書き込み
ｏｕｔｐ（ＢＡＳＥ＋０ｘ２Ｅ、ＤＡＤ15）// ＤＡチャンネル 15 データの書き込み

注　実際の各チャンネルへのＤＡ出力操作の前には、ステータスデータ中のビジーフラグがクリアされている事を毎回確認する必要があります。

ｏｕｔｐ（ＢＡＳＥ＋０ｘＥ、ｄｕｍｍｙ）// マニュアル同期更新

更新モードが単独動作の時は、各データ書き込み直後にＤＡ出力素子内臓のラッチが更新されます。同期動作の時は前３－５項で指定された要素（ソフトウェア処理或いは外部入力）により全チャンネルＤＡ素子内臓のラッチが同時に更新されます。

表３－７　【ＢＡＳＥ＋１０H】～【ＢＡＳＥ＋２ＥH】出力ポートの構成

ビット	ビット機能	適用	リセット時
B15 B14 B13 B12	未使用	無効データ	
B11 B10 B9 B8 B7 B6 B5 B4 B3 B2 B1 B0	D11(MSB) D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0	１２ビットＤＡデータ	

３－９　ステータスデータ取得、クリア

ｓｔｓ＝ｉｎｐ（ＢＡＳＥ＋０ｘ８）

本ボードに書き込まれたＤＡデータが（アイソレーターを通して）ＤＡ出力素子に転送中であることを示すＢＵＳＹフラグ､及びＵＰＤ＿ＩＮ端子に有効な割り込み信号が入力された事を示すＩＮＴフラグを読み込みます。

表３－８Ａ【ＢＡＳＥ＋０ｘ８】入力ポートの構成

ビット	各ビットの機能・意味	＝１の時	＝０の時	リセット時
Ｂ１５～Ｂ８	未使用			
Ｂ７	割り込み要求認識（ＩＮＴ）フラグ	割り込み要求発生	未発生・読込み済み	０
Ｂ６	外部入力（ＵＰＤ＿ＩＮ）現在状態	Ｈｉｇｈ	Ｌｏｗ	０
Ｂ５～Ｂ１	未使用			
Ｂ０	ＢＵＳＹフラグ	転送中	未転送・転送終了	０

補足説明

Ｂ７：当割り込み要求認識フラグは外部入力信号（ＵＰＤ＿ＩＮ）を割り込みとして許可している時、その指定極性エッジでセットされ同時に割り込み要求信号がＰＣＩバス上に出力されます。

Ｂ６：外部入力（ＵＰＤ＿ＩＮ）の現在値をモニターするビットです。割り込みを禁止し更にＤＡ出力モードを単独とした状態では汎用の１ビット入力として利用する事もできます。

Ｂ０：本機にＤＡ出力更新データを書き込む時はこのフラグが“０”でなければなりません。ＤＡ出力を連続して書き込む場合も書き込み出力前に必ずこのフラグが“０”であることを確認する必要があります。

ｏｕｔｐ（ＢＡＳＥ＋０ｘ８、０ｘ８０）// 割り込み要求認識フラグクリア

表３－８Ｂ【ＢＡＳＥ＋８Ｈ】出力ポートの構成

ビット	各ビットの機能・意味	＝１の時	＝０の時	リセット時
Ｂ１５～Ｂ８	未使用			
Ｂ７	割り込み要求認識（ＩＮＴ）フラグ処理	クリアする	何もしない	０
Ｂ６～Ｂ０	未使用			

３－10　汎用デジタル入出力

ｄｉｎ＝ｉｎｐ（ＢＡＳＥ＋０ｘＣ）// 汎用ＴＴＬ入力を取得

表３－９Ａ【ＢＡＳＥ＋０ｘＣＨ】入力ポートの構成

ビット	各ビットの機能・意味	＝１の時	＝０の時
Ｂ１５～Ｂ１	未使用		
Ｂ０	Ｉ０：汎用デジタル入力	Ｈｉｇｈ（開放）	Ｌｏｗ

注　この入力はソフト指定によって全アナログ出力の強制リセット（出力＝０Ｖ）にも使用できます。この場合
Ｉ０＝ＴＴＬレベルＨｉｇｈまたは開放の時：全アナログ出力はリセット（０Ｖ）状態
Ｉ０＝ＴＴＬレベルＬｏｗまたはグランドの時：全アナログ出力は通常状態

ｏｕｔｐ（ＢＡＳＥ＋０ｘＣ、ｄｏｕｔ）// 汎用ＴＴＬ出力の更新

表３－９Ｂ【ＢＡＳＥ＋０ｘＣＨ】出力ポートの構成

ビット	各ビットの機能・意味	＝１の時	＝０の時	リセット時
Ｂ１５～Ｂ１	未使用			
Ｂ０	Ｑ０：汎用デジタル出力	Ｈｉｇｈ（オフ）	Ｌｏｗ（オン）	０

表中の値はＳ－ＰＯＬジャンパーが“P”側に設定されている場合です。

【注１】出力素子はＳＮ７５４５２ＢＰを使用しています。（耐圧＝３０Ｖ、出力電流１００ｍＡmax
デバイスそのものは出力電流３００ｍＡまで使用する事が可能ですが、その時の出力電圧がＴＴＬローレベルの許容範囲を超えてしまうため１００ｍＡまでとします。）

【注２】電源投入またはハードウェアリセット直後の汎用デジタル出力の状態は“０”ですが、ソフト的な制御部リセット操作ではその状態は変わりません。

【注１】汎用デジタル出力の論理はボード上のジャンパーＳ－ＰＯＬによって任意に設定することができます。また出力回路については、ボード上のジャンパーＳ－ＰＵＰによってオープンコレクタ（Ｏ側）または１ＫΩのプルアップ（Ｕ側：標準）を選択できます。

第４章　Windows ハンドラー

４－０　ハードウェアとソフトウェアの連携

前章までで解説したハードウェアと、ユーザーアプリケーションとを接続する役目を果たしているのが、本章で説明するデバイスドライバーと呼ばれるソフトウェアになります。その動作はかなり複雑ですが、視覚的に簡略化した相互の関係を図４－０に示します。

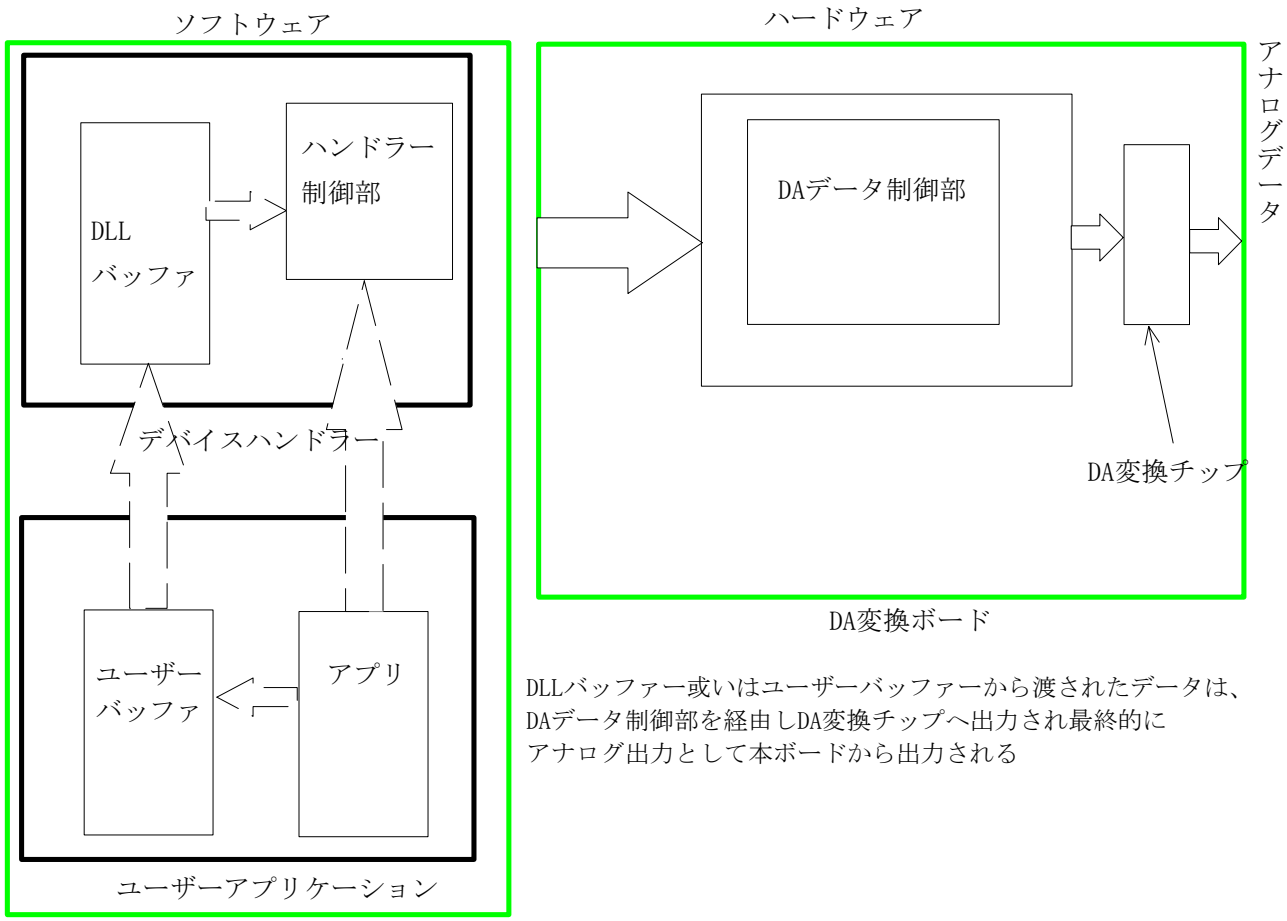


図４－０　ハードウェアｖｓユーザーアプリケーション相関図

DA 変換ボード内の全てのリソースは、ソフトウェアサイドから見ると入出力レジスタにマッピングされています。MSDOS の時代であれば、ユーザーアプリケーションサイドから直接入出力命令を発行する事により、ハードウェアの制御ができましたが、現在ではそのような操作はセキュリティ上の問題から禁止されており、これらレジスタの操作は、デバイスハンドラーというシステムサイドのソフトウェアを介在させて行うことになっています。

本章の以下の部分では、このデバイスドライバーを使用したハードウェア制御の方法を、順序だてて説明してゆく事になります。

尚、本デバイスドライバーはマイクロソフト製ビジュアルスタジオ 2008 を使用して製作されているため ビジュアルスタジオ 2008 用ランタイムライブラリが同時にインストールされます。

４－１　本製品用ソフトウェアの種類

本製品用のソフトウェアは、表４－１に示す３種類から構成されています。即ち、WindowsXP(32Bit)対応のデバイスドライバー、Windows7(32Bit)対応のデバイスドライバー及び Windows7(64Bit)対応のデバイスドライ

バーの３種類です。また、夫々のＯＳ用に専用ハンドラー並びに汎用ＩＯハンドラーが用意されています。

表４－１　ドライバー構成表

ＯＳ及びビット幅		Windows7 32Bit	Windows7 64Bit	WindowsXP 32Bit
パッケージ名称		Redist1616_x86.zip	Redist1616_x64.zip	Redist1616_xp.zip
専用ドライバー	ドライバー(DLL/Lib)	FCDA1616PCI_X86	FCDA1616PCI_X64	FCDA1616PCI_X86
	関数宣言	FCDA1616PCI.H	FCDA1616PCI.H	FCDA1616PCI.H
汎用ドライバー	ドライバー(DLL/Lib)	PORT7_X86	PORT7_X64	PORT7_X86
	関数宣言	PORT7.H	PORT7.H	PORT7.H

WindowsXP(32Bit)対応のドライバーは、アプリケーション移植時の利便性を考慮し用意されているものです。このドライバーを経由する事により、ドライバー更新→ＯＳ変更というステップを２段階に分割して実行できるように考慮されています。ドライバーそのものは Windows7(32Bit)版と同じものになっていますが、インストール方法の違いによりパッケージの構成が WindowsXP 対応の形になっています。
また全てのドライバー内の関数セット名称はマイクロサイエンスのドライバーと同一とし移植時の利便性を考慮しています。

４－２　ドライバーのインストール

４－２－１　WindowsXP ドライバーインストール及びアンインストール

WindowsXP 対応ドライバーをダウンロードし、その内容を全て解凍します。解凍場所は C ドライブのルートを指定して解凍します。解凍結果は以下のようになります。

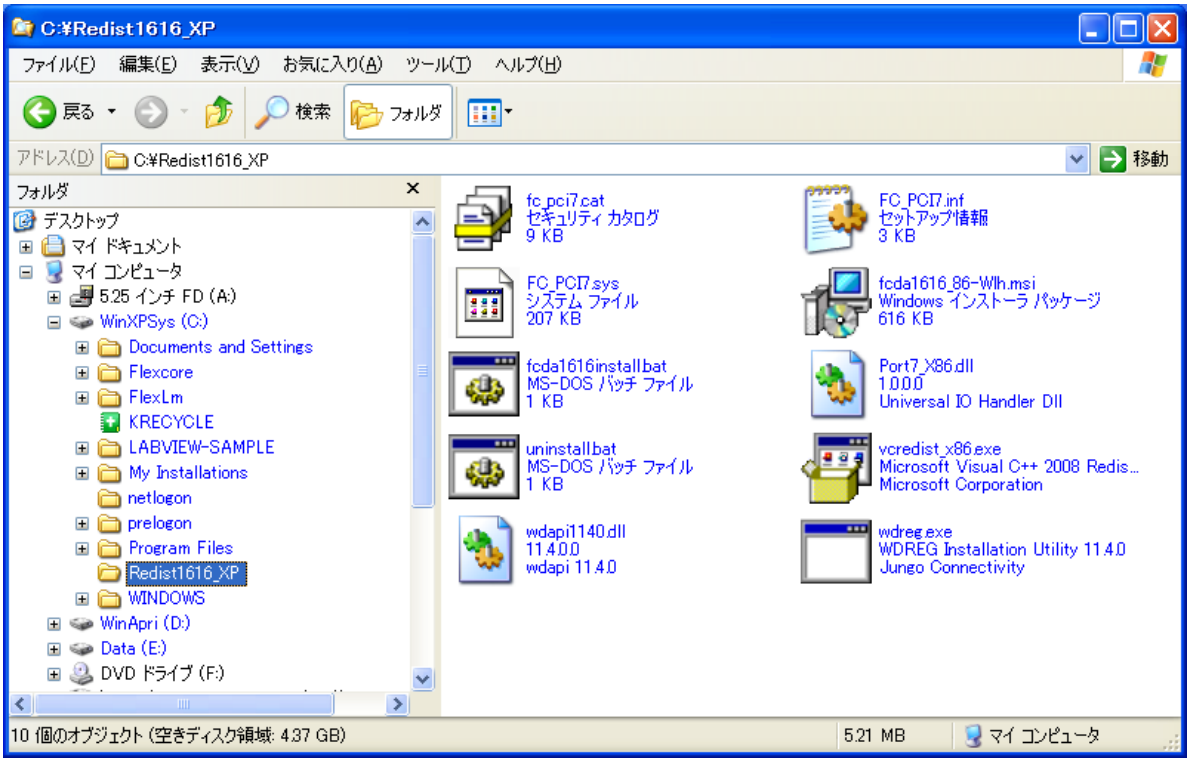


図４－２－１Ａ

デバイスインストールを開始する前に、旧ドライバーがインストールされていた場合、これを削除しなければなりません。そのためには、デバイスマネージャを開き、マイクロサイエンスドライバーが登録されているかどうかを確認します。

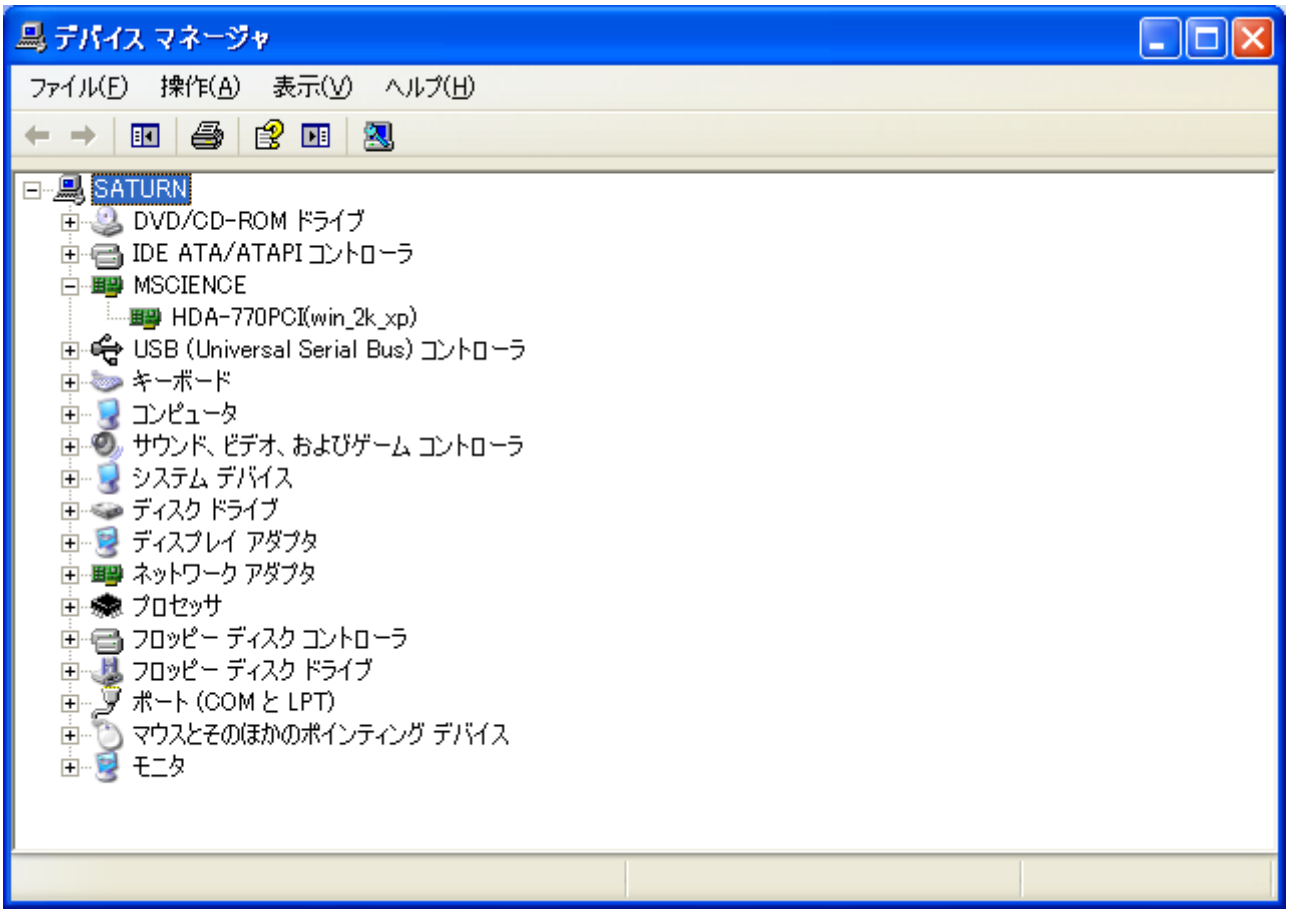


図４－２－１Ｂ

上図に示すようにMSCIENCEというエントリーが存在すれば、旧ドライバーがインストールされています。
ドライバー削除は、

1　まずWindows¥Infフォルダの中からOem*.Infファイルを検索します。検索条件としては、文字列として“MSCIENCE”が含まれているものを条件とします。検索されたファイルをテキストエディターでマイクロサイエンスのものである事を確認し、Oemxx.Inf（xxは上記の条件に合致した番号です）とOemxx.Pnfを削除します。

2　次にWindows¥System32フォルダの中から、DMS_PCI.SysとDMS_PCI.Infを探し同じように削除します。

3　最後に、図４－２－１Ｂで示すマイクロサイエンス製ドライバー（例ではHAD-770PCIです）を右クリックし削除します。

ここまできた段階で一旦パソコンの電源を落とし、ボードを取り外します。
再度パソコンの電源を投入し、図４－２－１Ａで示されるフォルダの中から“Fcda1616PciInstall.Bat”ファイルを右クリックし“開く”（実行する）を選択します。すると、まず図４－２－１Ｃで示されるウィンドーが開きます。（次ページ）

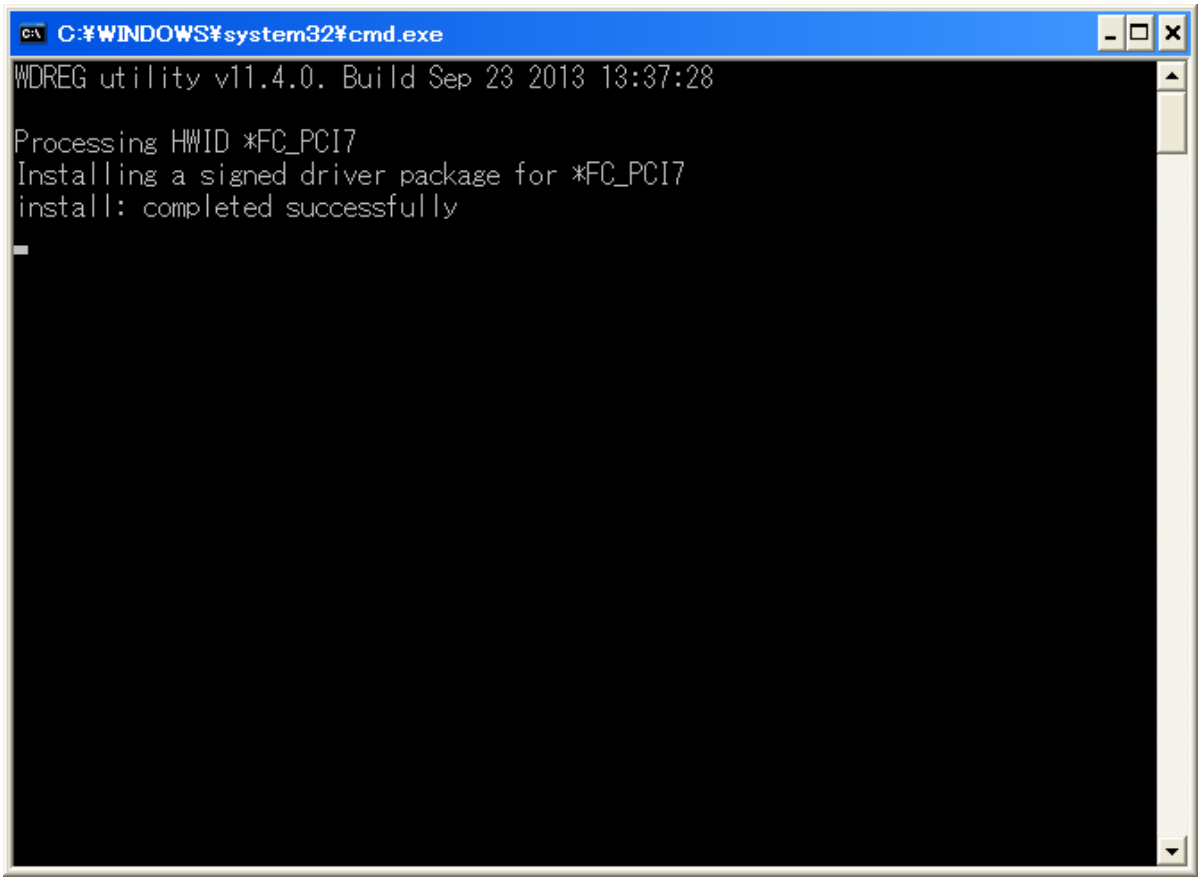
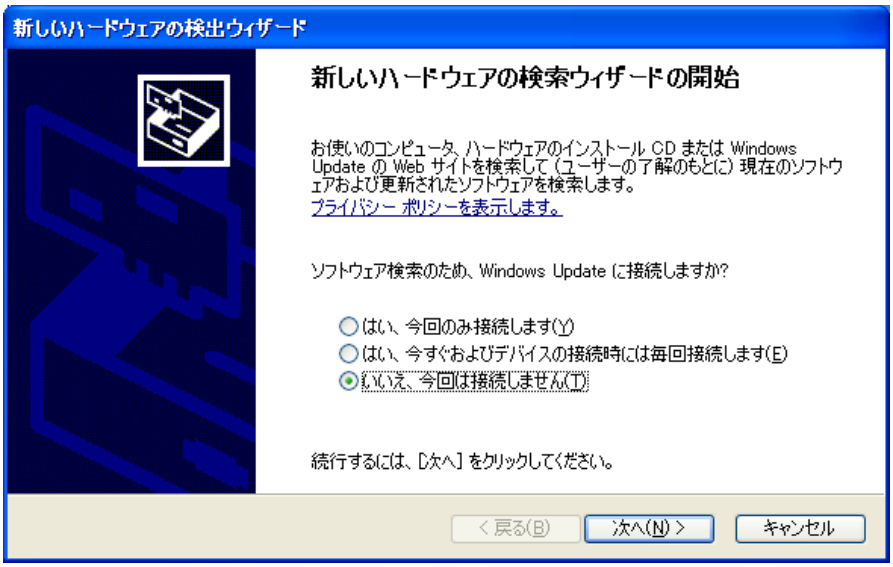
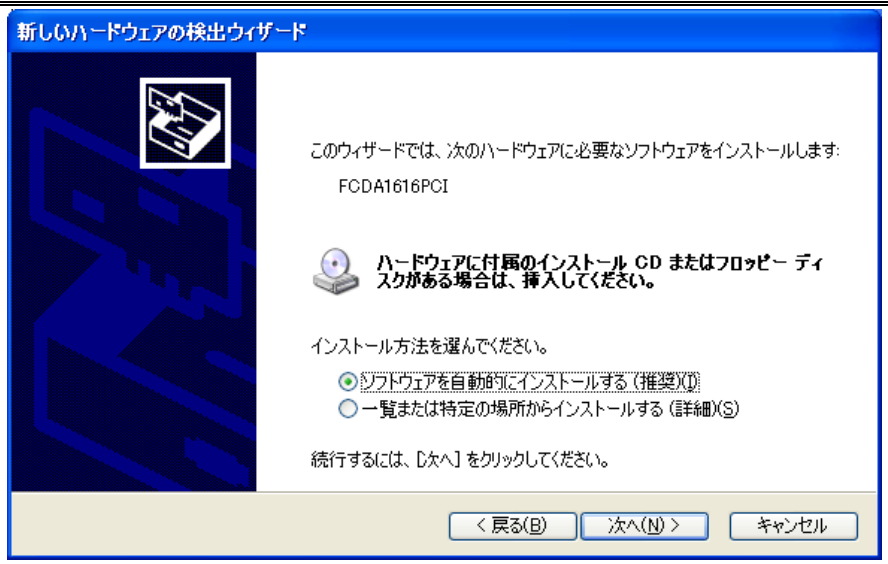


図４－２－１C

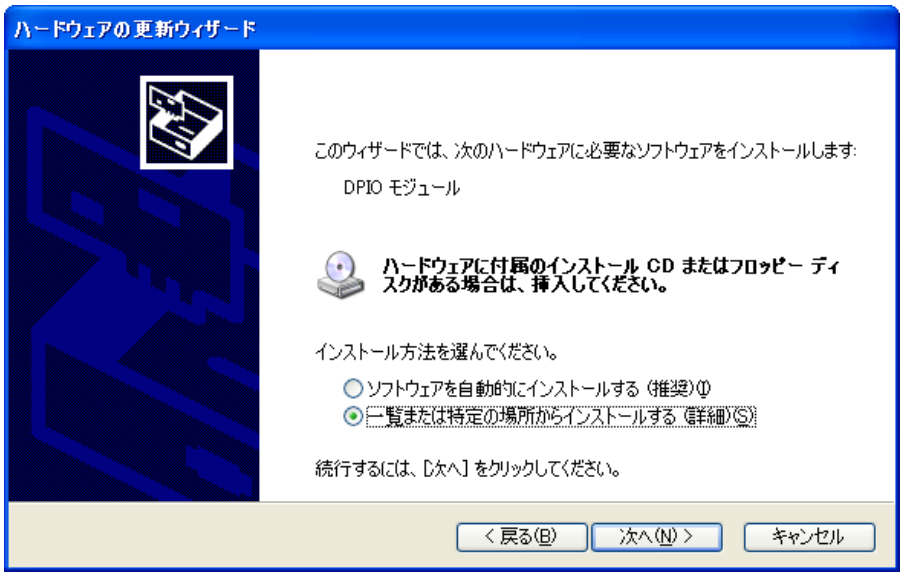
そして暫くすると、インストール作業が終了します。そこで再度パソコンの電源を遮断し FCDA1616PCI ボードを再度挿入し電源を投入します。システムが立ち上がるとハードウェアウィザードが立ち上がりますから、



WindowsUpdate への接続をせず、“次へ“ボタンをクリックします。そして次ページの例のように”次のハードウェア“として FCDA1616PCI が表示された場合には



推奨設定のまま、“次へ”ボタンをクリックします。すると必要なDllファイル群がシステムフォルダにコピーされインストール作業は終了します。
また、上記ウィンドーで次のハードウェアとして“DPIOモジュール”が表示された場合には、



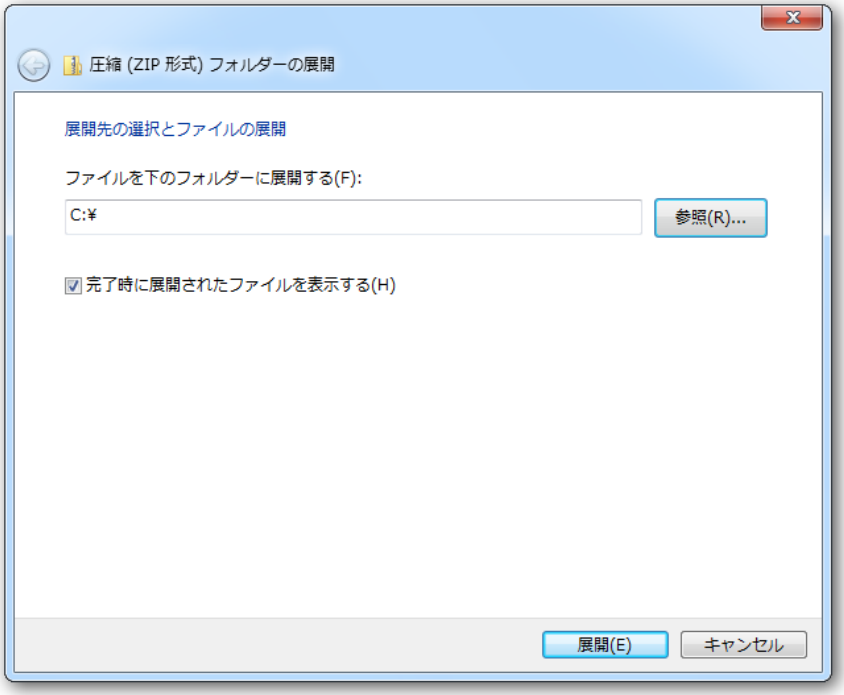
“一覧または特定の場所からインストールする”オプションを選択し次の画面で“C:¥Redist1616_XP”フォルダを指定してインストールを行います。
ドライバーを削除する場合は、図４－２－１Ａで“Uninstall.Bat”ファイルを実行することで自動的にドライバーが削除されます。

４－２－２　Windows７へのインストール

本製品用のソフトウェアは弊社ホームページ内ダウンロードコーナーにアップロードされています。その中から、実際に御使用になるＯＳとマッチしたドライバーセットをダウンロードして頂き本ボードを取り付ける前にデバイスドライバーをインストールしていただきます。

実際のインストール方法は以下の通りです。

・　本ボードをパソコンにセットアップする前に、弊社ホームページよりご使用のＯＳ用デバイスドライバーキットをダウンロードして頂きパソコンの中に解凍します。解凍場所は、Ｃドライブのルートとします。（本ドライバーキットには、デバイスハンドラーが含まれています。）



前ページの“展開”ボタンをクリックするとドライバーセットがＣドライブ内に展開されます。内容は図４－２－２Ａを参照してください。

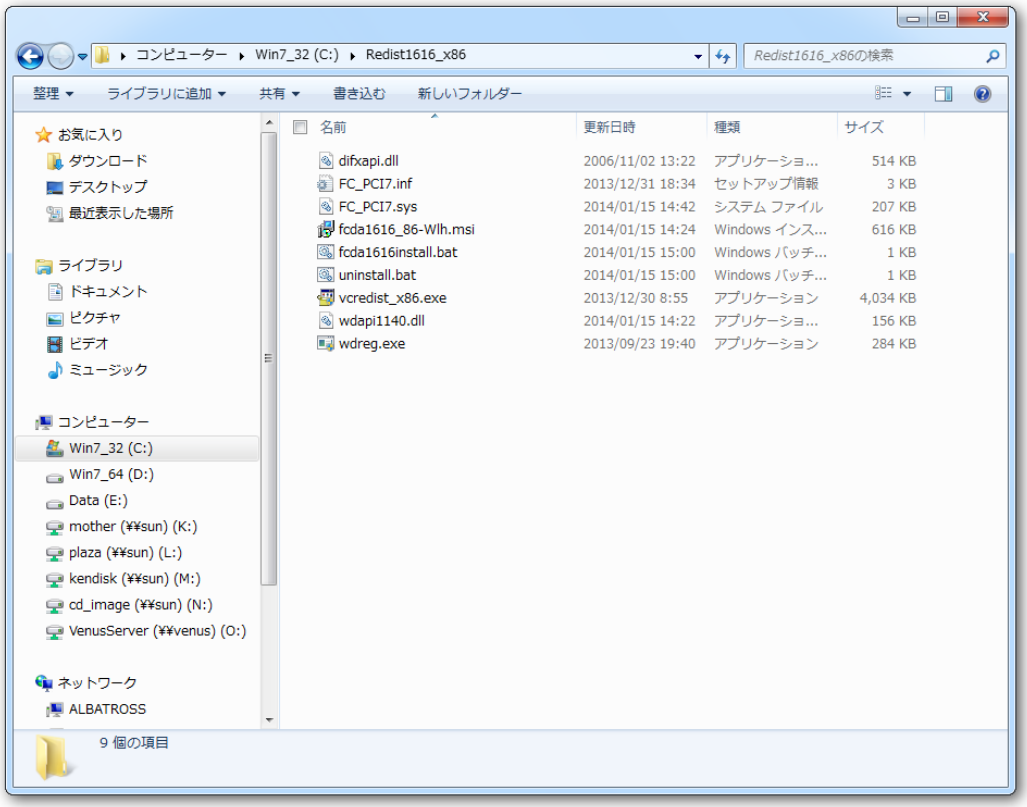
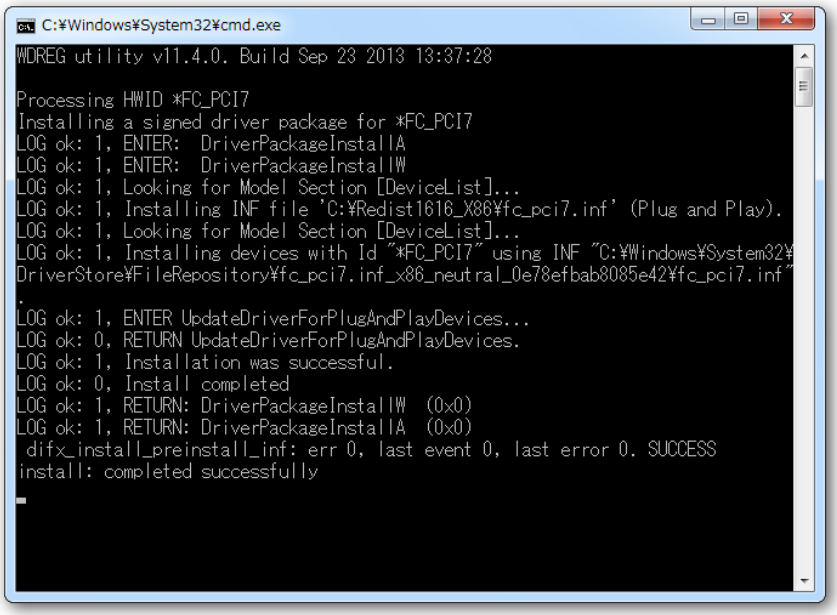


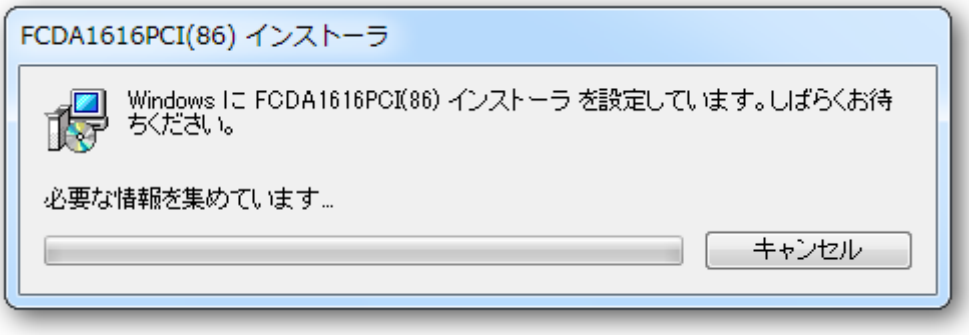
図４－２－２Ａ

この中で“Fcda1616Install.Bat”がデバイスドライバーインストーラーです。***また、この時点で、本ボードは、まだパソコンには取り付けないで***下さい。この順序が守られない場合、正常なインストールは保証されません。このデバイスドライバーインストーラーを***管理者として***実行（右クリックメニューから選択）すると次のウィンド

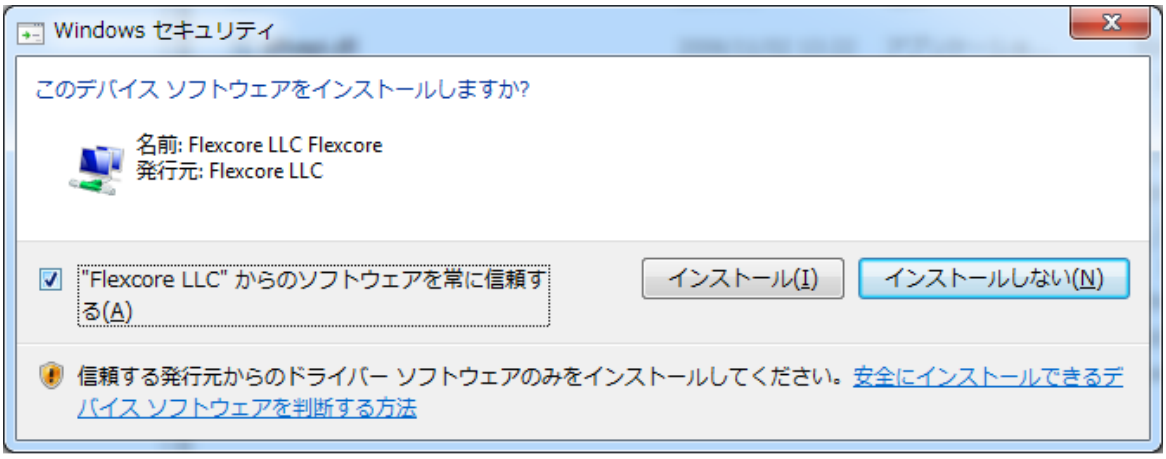
ーが開きます。（ただ実行とした場合は、ウィンドーが瞬間的に終了しデバイスドライバーのインストールはできません）



更に暫く待っていると、次のウィンドーが開きます。

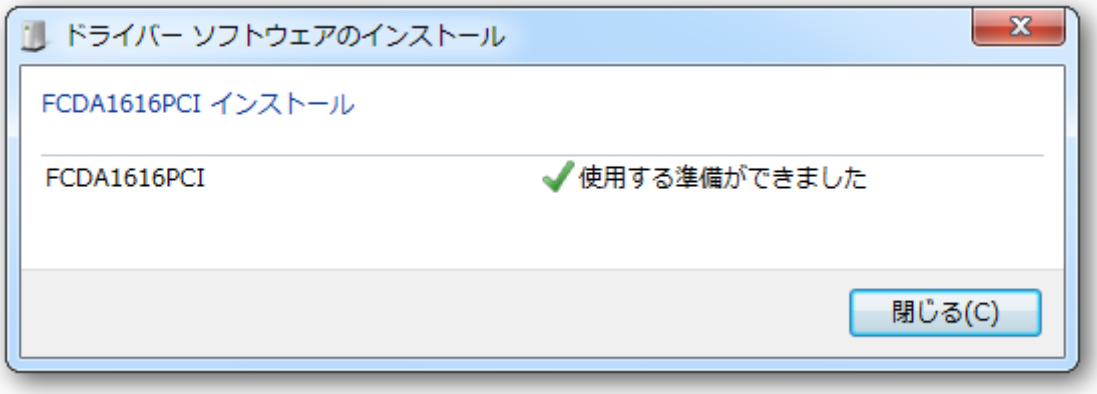


更に待っていると次のウィンドーが開きます。



ここでは、“Ｆｌｅｘｃｏｒｅ　ＬＬＣ”からのソフトウェアを常に信頼するという項目にチェックを入れ“インストール”ボタンをクリックします。そして暫く待っているとインストール作業が終了します。
この後、一旦パソコンの電源を落とし、パソコン内部のＰＣＩスロット或いはＰＣＩ拡張ユニットに本ボードを挿入し再度パソコンの電源を投入します。ＯＳが立ち上がるとハードウェアの検出が行われ、少し待っていると画面

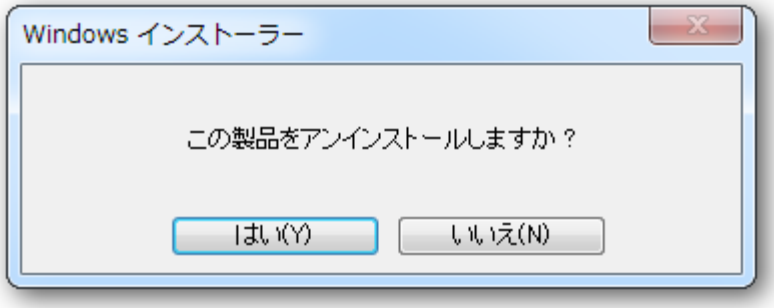
下部のタスクバーの所に、デバイスドライバーが正常にインストールされたというメッセージが表示されます。



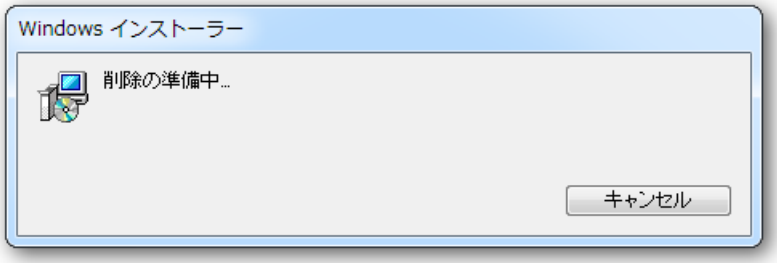
これで本ユニットのインストールは完了しました。

４－３　Windows７ドライバーのアンインストール

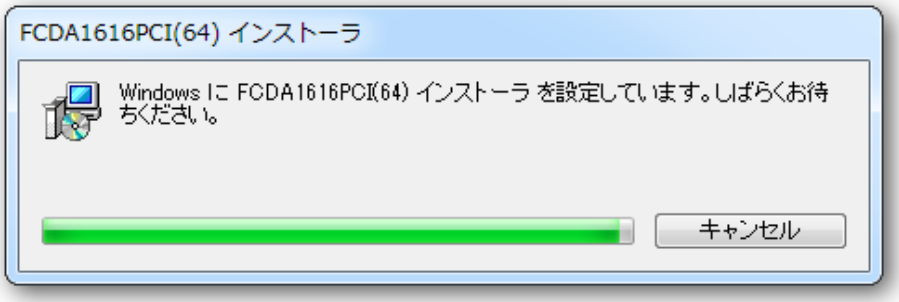
この方式でインストールしたドライバーはドライバー更新時などにドライバーをアンインストールする際にも簡単に行う事ができます。まず、４－２－１項で説明したデバイスドライバー解凍フォルダへ移動し、この中にある“Uninstall.Bat”ファイルを実行します。（ここでも*管理者として*実行する必要があります。）



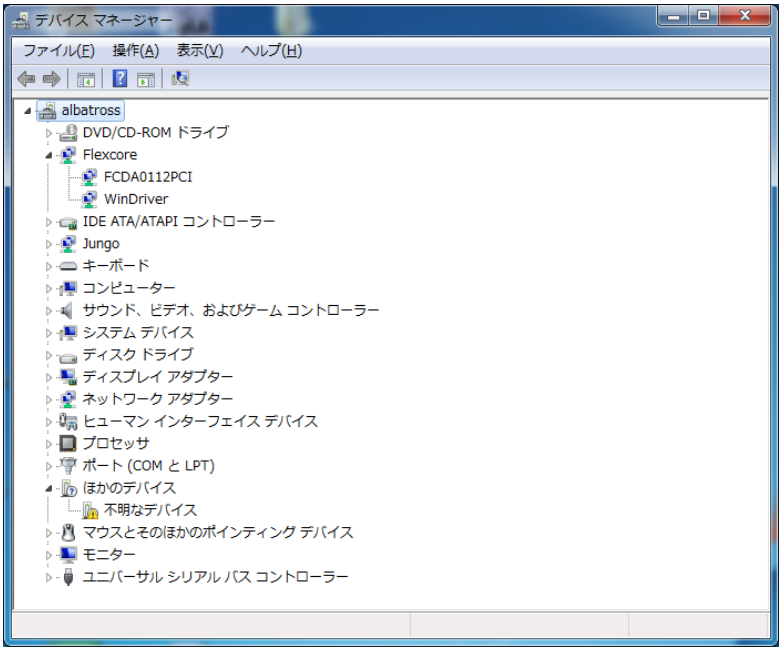
というメッセージが現れますので“はい”をクリックします。すると



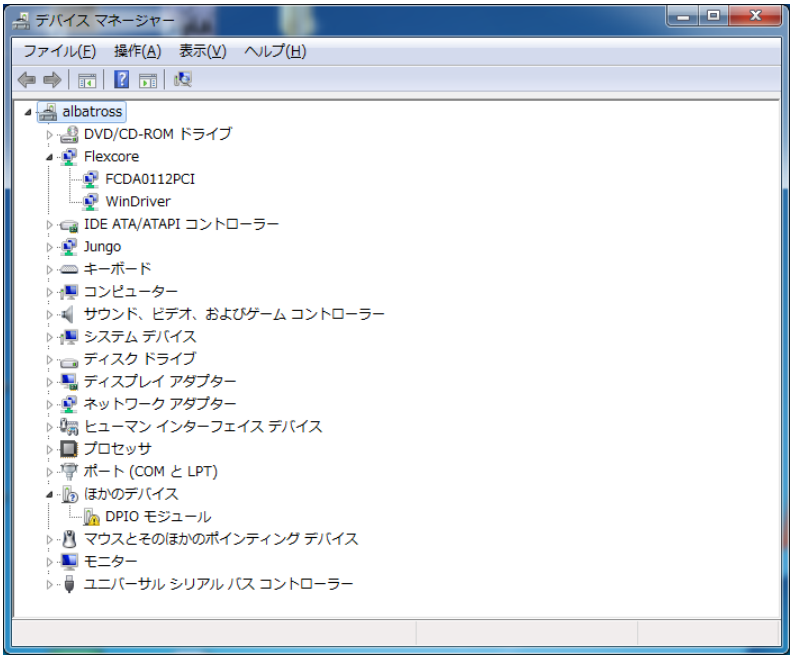
というウィンドーに変わり、更に待っていると



というウィンドーに変わり、暫く待っているとデバイスドライバーが削除されます。
この段階で、スタートー＞コントロールパネルと進み、デバイスマネージャを開くと、下図に示すように不明なデバイスが表示されています。(同図下から４行目)
他方、上から４行目にはアンインストールされていないボード情報も存在しています。



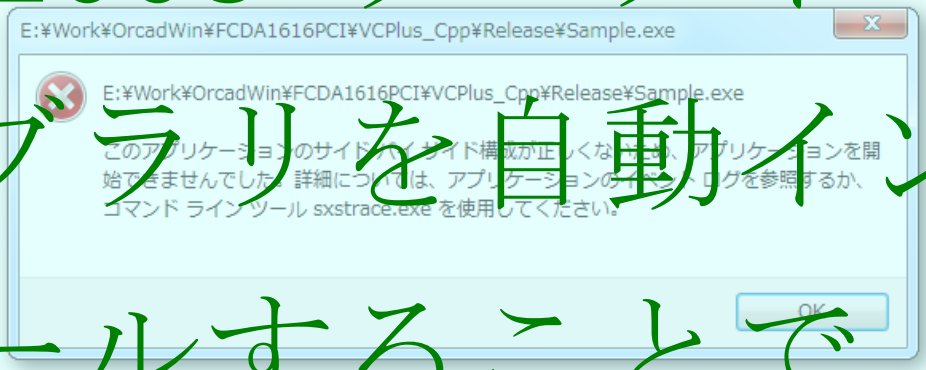
そこで、この不明なデバイスを右クリックし、表示されるドロップダウンリストから“デバイスの削除”を選び実行します。その結果、次ページに示すように不明なデバイスが、ドライバー未インストール時と同様、ほかのデバイスの中にＤＰＩＯモジュールが存在する状態に変化します。(この表示を得るためには、トップエントリ：同図では albatross と表示されているエントリーを右クリックし“ハードウェア変更のスキャン”を実行する必要があります)
この状態に到達すると、パソコンの電源を遮断し、本ボードを取り外す事ができます。
　特に、バージョンの異なるドライバー（例えば Windows8 対応のドライバー）や、他メーカーのドライバーに切り替える時には、下の図で示されるように、本ボードが DPIO モジュールという状態になっている必要があります。



この状態で、電源遮断後、本ボードをシステムから取り除くと、４－２項　ドライバーインストール直前の状態に復帰している事になります。

~~４－４　動作確認プログラムについて（補足）~~

動作確認プログラムを実際に動作させると、以下のようなメッセージが出て動作しない事があります。これは、Windows7若しくはWindowsXPにおいてＣ＋ランタイムが標準でインストールされていないためです。



Windows7の場合

WindowsXPの場合

弊社で作成した動作確認プログラムはＭＳＶＣ＋２００８でコンパイルされておりますので、専用のランタイムをインストールする事により、この状況を回避する事ができます。必要となるランタイムは、ドライバーパッケージの中に同梱しておりますので、こちらを実行することでランタイムのインストールができます。ファイル名称はｖｃｒｅｄｉｓｔ_ｘ８６.ｅｘｅ（Ｘ８６環境）又はｖｃｒｅｄｉｓｔ_ｘ６４.ｅｘｅ（Ｘ６４環境）になります。

若しくは、（まだＣ＋環境をインストールされていない場合）当該パソコンにＭＳＶＣ＋２００８をインストールする事でも対処する事ができます。

VS2008 ランタイムライブラリを自動インストールすることで、本注意事項は不要となったため削除

４－５　関数仕様

連番	関数名	機能・動作	主なパラメータ等
【１】	DA770_Open_DASys	ボード、ハンドラーの初期化	
【２】	DA770_Reset_Board	ボードのリセット	
【３】	DA770_Set_SampMode	DA 出力モードの設定	
【４】	DA770_Set_Range	DA 出力範囲、コードの指定	
【５】	DA770_Out_Prompt	DA 即時更新出力	
【６】	DA770_Get_Status	ボードステータスの取得	
【７】	DA770_Close_DASys	本ハンドラーの終了	
【８】	DA770_Out_Aux	汎用デジタル出力	
【９】	DA770_Inp_Aux	汎用デジタル入力	
【10】	DA770_Force_0v	デジタル入力による強制０Ｖ出力	
【11】	DA770_Sync_Update	全チャンネル同期更新命令	
【12】	DA770_Get_Libver	バージョン情報取得	

【１】ボード、ハンドラーの初期化

int __stdcall DA770_Open_DASys (int Num_Board)	
引数	Num_Board:使用ボード枚数
戻り値	正常終了時：０ エラー時：エラーコード（負の値）
機能・動作	プラグアンドプレイで設定された各ボードのＩ／Ｏアドレス、及び割り込みレベルを認識すると同時にボードリセット、その他ハンドラー内の各種テーブルやデータバッファ等を初期化する。

【２】ボードのリセット

int __stdcall DA770_Reset_Board (void)	
引数	なし
戻り値	正常終了時：０ エラー時：エラーコード（負の値）
機能・操作	全ての制御レジスタは初期状態に戻る。（但し汎用デジタル出力のみは変化しない） アナログ出力は０Ｖに戻る

【３】DA 出力モードの設定

int __stdcall DA770_Set_SampMode(int upd_mode)	
引数	upd_mode：ＤＡ出力更新モード指定（０：単独更新、１：同期更新（全ボード全チャンネルが一斉に更新））
戻り値	正常終了時：０ エラー時：エラーコード（負の値）
機能・操作	ＤＡ出力更新タイミング条件を設定する

【４】DA 出力範囲、コードの指定

int __stdcall DA770_Set_Range(int board_no, int ch_no, int range, int data_code)	
引数	board_no：設定対象ボード番号（０～１５） ch_no：設定対象ＤＡ出力チャンネル番号（０～１５） range：ＤＡ出力範囲（０：０～１０Ｖ、１：０～５Ｖ、２：±１０Ｖ、３：±５Ｖ） data_code：ＤＡ出力データコード（０：バイナリ、１：２の補数）
戻り値	正常終了時：０ エラー時：エラーコード（負の値）
機能・操作	各ボードの各チャンネル毎に DA 出力範囲を設定します。 ＤＡ出力コードは全ボード全チャンネルに同一コードを指定して下さい。 （異なるコードを指定すると最後に指定したコードが全ボード全チャンネルに適用されます）

【５】DA 即時更新出力

int __stdcall DA770_Out_Prompt(int board_no, int ch_no, WORD upd_prompt)	
引数	board_no：設定対象ボード番号（０～１５） ch_no：設定対象ＤＡ出力チャンネル番号（０～１５） upd_prompt：ＤＡ出力データ（整数：digit）
戻り値	正常終了時：０ エラー時：エラーコード（負の値）
機能・操作	ＤＡ出力更新モードが単独更新の場合は指定チャンネルのみが即時更新されます。 ＤＡ出力更新モードが同期更新の場合は、各ボード各チャンネルについて本関数で更新データを書き込んでおき、最後に全ｃｈ同期更新命令を実行すると全チャンネルが同一タイミングで更新出力されます。

【６】ボードステータスの取得

int __stdcall DA770_Get_Status (int *busy, int *status)	
引数	busy：ボード内部のＤＡデータ転送フラグ（０：未転送／転送終了、１：転送中） status：ボードステータス（１ワードの生データ）
戻り値	正常終了時：０ エラー時：エラーコード（負の値）
機能・操作	マスターボードのステータスを取得します。

【７】本ハンドラーの終了

int __stdcall DA770_Close_DASys (void)	
引数	なし
戻り値	正常終了時：０ エラー時：エラーコード（負の値）
機能・操作	本ハンドラーを終了します。確保したメモリ領域の開放等、開始前の状態に戻します。但しＤＡ出力最終状態を維持するためにボードのリセットは行いません。

【８】汎用デジタル出力

int __stdcall DA770_Out_Aux(int board_no, int out_data)	
引数	board_no：出力対象ボード番号（０～１５） out_data：１ビット汎用デジタル出力データ
戻り値	正常終了時：０ エラー時：エラーコード（負の値）
機能・操作	ＴＴＬレベルの１ビット汎用デジタル出力を更新する。当出力ポートは電源投入時（ハードウェアリセット時）０となるが初期設定、ボードリセット、ハンドラー終了などを実行しても変化しません。また出力論理はボード上の SW-PL スイッチによって行います。

【９】汎用デジタル入力

int __stdcall DA770_Inp_Aux(int board_no)	
引数	board_no：出力対象ボード番号（０～１５）
戻り値	正常終了時：１ビット汎用デジタル入力データ エラー時：エラーコード（負の値）
機能・操作	ＴＴＬレベルの１ビット汎用デジタル入力を読み込む。

【10】デジタル入力による強制０Ｖ出力

int __stdcall DA770_Force_0v(int board_no, int set)	
引数	board_no：出力対象ボード番号（０～１５） set：設定の有無（０：非設定、１：設定）
戻り値	正常終了時：０ エラー時：エラーコード（負の値）
機能・操作	ボードの汎用デジタル入力ビット０を緊急入力とし、その入力がハイ（または開放）となるとＤＡ出力を強制的に０Ｖとする機能を設定するかどうかを処理する関数です。

【11】全チャンネル同期更新命令

int __stdcall DA770_Sync_Update(void)	
引数	なし
戻り値	正常終了時：０ エラー時：エラーコード（負の値）
機能・操作	同期更新モードの時、本関数の実行によって全ボード全チャンネルのＤＡ出力が同時更新されます。

【12】バージョン情報取得

int __stdcall DA770_Get_Libver(int ver)	
引数	int：０（戻り値はメジャーバージョン番号＋マイナーバージョン番号） １（戻り値はメジャーバージョン番号） ２（戻り値はマイナーバージョン番号）
戻り値	正常終了時：本ハンドラーのバージョン番号 エラー時：エラーコード（負の値）
機能・操作	本ハンドラーのバージョン情報を取得する関数です。 例えば、バージョンが１．０１の場合、本関数をｖｅｒ＝０として実行すると 戻り値は０ｘ０１０１となります。

エラーコード一覧

戻り値	不具合の内容	適用関数等
－１	PCI ライブラリが見つからない	DA770_Open_DASys 関数
－２	デバッグオプションが無効だった	同上
－３	PCI ボードが見つからなかった	同上
－４	FCDA1616PCI ボードが見つからなかった	同上
－５	マスターユニットが見つからなかった	同上
－６	SW-BN スイッチが重複している	同上
－７	ボード ID が間違っていた	同上
－８	更新モード指定が間違っていた	DA770_Set_SampMode 関数
－９	ボード番号指定がシステム上限枚数を超えている	
－１０	チャンネル指定が物理チャンネル数を越えている	チャンネル数を指定する関数
－１１	レンジ指定が間違っている	DA770_Set_Range 関数
－１２	データコード指定が間違っている	同上
－１３	指定したユニット番号のボードが見つからない	ドライバー内部関数
－１４	出力データが許容範囲外だった	DA770_Out_Aux 関数
－１５	強制０Ｖ出力指定データが許容範囲外だった	DA770_Force_0v 関数
－１６	バージョン要求変数が指定範囲外だった	DA770_Get_Libver 関数
－１７	指定されたボード数と実際のボード枚数が異なっていた	DA770_Open_DASys 関数

第５章　汎用デバイスドライバー仕様書

本仕様書は、フレックスコア製 PCI ボードを制御するために必要となる汎用デバイスドライバーの内容について記述したものです。本デバイスドライバーは、最大で 16 台までの PCI ボードを同時に制御する事が可能ですが、基本的な仕組みは、ドライバーオープン及びドライバースタートから始まり、夫々の PCI ボードリソースを、Ｉ／O制御レジスタへのアクセスにより制御し、全ての制御が終了した段階でドライバーストップ、更にドライバークローズ処理を行って制御全てを終了するところまでを行います。
基本的なリソース制御については、バイト制御、バイトブロック制御、ワード制御、ワードブロック制御、更にダブルワード制御及びダブルワードブロック制御の１２種類（各々リード・ライト処理がある）が定義されています。実際には、本仕様書脱稿の段階でダブルワード制御を行うレジスタを装備したボードは存在しませんが、将来的にそのようなボードが成立する可能性も充分考えられるため、ハンドラー仕様書として内容を整備しています。

本ドライバーは、Windows7 等での PCI ボードインストールとの関係上、対応する専用ドライバーインストール時に共にインストールされるよう設定されています。（即ち専用のインストーラーは存在しません。）

また、本ドライバーは、上記の如く単純なＩ／Ｏレジスタアクセスのみから構成されている為、基本的に無サポートとします。また全てのデータ処理（バイト／ワード／ダブルワード：リード／ライト）については、クリティカルセクションによって排他処理が保証されています。

※　クリティカルセクションについては、特にマルチコアＣＰＵ等の環境で本ドライバーを使用する際に必要となるものですが、ここではその詳細については記述しません。詳細については、マイクロソフト製ビジュアルスタジオ等に詳細な記述があるのでこれを参照してください。

ハンドラー及び関連ファイル名称一覧表

ＯＳ及びビット幅	Windows7　32 ビット	Windows7　64 ビット	WindowsXP　32 ビット
ドライバー名称	PORT7_X86.Dll	PORT7_X64.Dll	PORT7_X86.Dll
インポートライブラリ名称	PORT7_X86.Lib	PORT7_X64.Lib	PORT7_X86.Lib
関数インデックス	PORT7.H	PORT7.H	PORT7.H

上記ドライバー３種類の内、WindowsXP32 ビット用ドライバーについては、システム移行動作確認用としてのみ使用する事を前提として公開しています。このドライバーは、WindowsXP システムで動作していたアプリを Windows7 へ移行する際に、その操作を段階的に（つまり OS の変更を行うこと無しに）新しいドライバーでの動作評価を行うことができるよう用意されているものです。（ハンドラーの内容としては Windows７　３２ビット版と全く同じ構成になっています。）

以下に記述する仕様は、ある固有ベンダーID に従った PCI ボード群に対する汎用デバイスドライバーの仕様書です。ベンダーID は固有の値（現状０ｘ１３ＦＤ）でなければなりません。

【1】ドライバーオープン

int __stdcall Open_Driver(int *device_id);	
引数	device_id:各ボードのデバイス ID をセットする配列（ユーザー側にて確保:最大 16 個）
戻り値	正常終了時：0 エラー発生時：エラーコード（負の値）
機能・動作	ハンドラー内部の初期化を行い、システム内で検出されたそれぞれのボードのデバイス ID を報告する。

【2】ドライバースタート

int __stdcall Start_Driver(int board_no, int base_no, int *resorce_no);	
引数	board_no: ドライバーオープンで報告された、デバイス ID を格納している配列の要素番号であり、尚且つシステム内で検索され発見された該当ボードの発見順序を表す変数である base_no:ＩＯリソース番号（２固定） resource_no:当汎用ハンドラー内で確定したリソースを指し示す要素番号
戻り値	正常終了時：0 エラー：エラーコード（負の値）
機能・動作	ドライバーオープンで検出されたボード群の内 board_no で指定したボードと通信を開始する。

【3】バイトリード

int __stdcall ReadByte(int resource_no, int offset);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス
戻り値	正常終了時：読み込んだバイト値 エラー発生時：エラーコード（負の値）
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタの値を読み込みユーザーへ返す関数（戻り値＝バイト）

【4】バイトブロックリード

int __stdcall ReadByteBlock(int resource_no, int offset, int count, BYTE *pbuf);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス count:読込みを行うバイト数 pbuf:読み込んだレジスタからの値を保存する先頭アドレス（ポインター）
戻り値	正常終了時：0 エラー発生時：エラーコード（負の値）
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタの値を count 回数読み込みユーザーが用意したデータバッファ（pbuf）へ返す関数である。pbuf データバッファのサイズは count（バイト）必要である。

【5】ワードリード

int __stdcall ReadWord(int resource_no, int offset);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス
戻り値	正常終了時：読み込んだワード値 エラー発生時：エラーコード（負の値）
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタの値を読み込みユーザーへ返す関数

【6】ワードブロックリード

int __stdcall ReadWordBlock(int resource_no, int offset, int count, WORD *pbuf);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス count:読込みを行うワード数 pbuf:読み込んだレジスタからの値を保存する先頭アドレス（ポインター）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	board_no で示されるボードの offset アドレスに存在するレジスタの値を count 回数読み込みユーザーが用意したデータバッファ（ret_value）へ返す関数である。Ret_value データバッファのサイズは count×２（バイト）必要である。

【7】ダブルワードリード

int __stdcall ReadDWord(int resource_no, int offset);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス
戻り値	正常終了時：読み込んだダブルワード値 エラー発生時：エラーコード（負の値）
機能・動作	board_no で示されるボードの offset アドレスに存在するレジスタの値を読み込みユーザーへ返す関数

【8】ダブルワードブロックリード

int __stdcall ReadDWordBlock(int resource_no, int offset, int count, UINT32 *pbuf);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス count:読込みを行うワード数 pbuf:読み込んだレジスタからの値を保存する先頭アドレス（ポインター）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタの値を count 回数読み込みユーザーが用意したデータバッファ（pbuf）へ返す関数である。pbuf データバッファのサイズは count×４必要である。

【9】バイトライト

int __stdcall WriteByte(int resource_no, int offset, BYTE data);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス data:レジスタへ書き込む値
戻り値	正常終了時：0 エラー発生時：エラーコード（負の値）
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタへ data を書き込む関数

【10】バイトブロックライト

int __stdcall WriteByteBlock(int resource_no, int offset, int count, BYTE *pbuf);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス count:読込みを行うワード数 pbuf:読み込んだレジスタからの値を保存する先頭アドレス（ポインター）
戻り値	正常終了時：0 エラー発生時：エラーコード（負の値）
機能・動作	board_no で示されるボードの offset アドレスに存在するレジスタへ pbuf バッファーから count 回数書き込みを行う関数である。Pbuf データバッファのサイズは count（バイト）必要である。

【11】ワードライト

int __stdcall WriteWord(int resource_no, int offset, WORD data);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス data:レジスタへ書き込む値
戻り値	正常終了時：0 エラー発生時：エラーコード（負の値）
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタへ data に保存されているデータを書き込む関数

【12】ワードブロックライト

int __stdcall WriteWordBlock(int resource_no, int offset, int count, WORD *pbuf);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス count:書き込みデータ数 pbuf:レジスタへ書き込む値を保存しているアドレス（ポインター）
戻り値	正常終了時：0 エラー発生時：エラーコード（負の値）
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタへ pbuf に保存されているデータを書き込む関数。pbuf データバッファのサイズは count×２（バイト）必要である。

【13】ダブルワードライト

int __stdcall WriteDWord(int resource_no, int offset, UINT32 data);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス data:レジスタへ書き込む値
戻り値	正常終了時：0 エラー発生時：エラーコード（負の値）
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタへ data に保存されているデータを書き込む関数

【14】ダブルワードブロックライト

int __stdcall WriteDWordBlock(int resource_no, int offset, int count, UINT32 *pbuf);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス count:書き込みデータ数 pbuf:レジスタへ書き込む値を保存しているアドレス（ポインター）
戻り値	正常終了時：0 エラー発生時：エラーコード（負の値）
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタへ pbuf に保存されているデータを書き込む関数。pbuf データバッファのサイズは count×４（バイト）必要である。

【15】ドライバーストップ

int __stdcall Stop_Driver(int resource_no);	
引数	resource_no: ドライバースタート関数から返されたボードリソースインデックス
戻り値	正常終了時：0 エラー発生時：エラーコード（負の値）
機能・動作	全てのユニットで使用していたメモリリソースを開放し通信を終了する。

【16】ドライバークローズ

int __stdcall Close_Driver(void);	
引数	なし
戻り値	正常終了時：0 エラー発生時：エラーコード（負の値）
機能・動作	全てのボードについてハンドルを開放する。

エラーコード

戻り値	内容
－１	必要な PCI ライブラリが見つからない
－２	ライブラリのデバッグオプションが無効だった
－３	PCI ボードが見つからなかった
－４	ボード番号指定がシステム上の枚数を上回っている
－５	オープンしていないボードをクローズしようとした
－６	定義されていないベースナンバーを参照しようとした
－７	オープンしていないボードに対してアクセスが行われた
－８	リソース範囲不適

第６章　保守、その他

６－１　故障、トラブル等の原因と対処

本ボードは、ＤＯＳ／Ｖパソコン＋拡張ボックスのシステム構成で全数検査の上出荷されています。またお手元での動作確認方法は１－６項に記載されています。動作に不具合があるときは、以下の諸点を再度点検して下さい。それでも不明な時は、システム構成などの動作条件を明示し弊社宛Ｆａｘ若しくはＥｍａｉｌをお願いします。基本的に速度重視の時はＥｍａｉｌが、内容が複雑な時はＦａｘがお勧めです。

再点検・確認ポイント

・Ｉ／Ｏアドレス　　ボードインストールは成功したか、デバイスマネージャ上で？等の不審なマークはないか

・アナログ出力　　負荷：抵抗値５ＫΩ／チャンネル以上、容量１０００ｐＦ／チャンネル以下の条件を満たしているか

・デジタル入出力　　接続先の電圧レベルはＴＴＬレベルコンパチか？

６－２　修理のときは

本製品使用上の質問･トラブル対応･故障修理などは入手経路の如何にかかわらず､弊社宛に直接御相談下さい。商社等を経由されても弊社として問題はありませんが、情報交換の密度、修理完了までの時間などの面で多少不利になるのではないかと思われます。
また、導入当初からの不具合については、誤解や情報不足による事が多いので、事前にご相談下さい。
無償修理と有償修理の場合分けにつきましては以下の通りとなります。

納入後１年間は自然故障､及び弊社製造上の問題に起因した事が明らかな故障製品に対して無償修理を行います。
但し保証期間中であっても、次の場合は有償修理となります。

ア　取扱上の不注意、誤用による故障および損傷
イ　弊社以外での修理、改造、分解掃除等による故障および損傷
ウ　泥、砂、水などのかぶり、落下、衝撃等が原因で発生した故障および損傷
エ　火災、地震、水害、落雷その他の天変地異、公害や異常電圧による故障および損傷
オ　保管上の不備（高温多湿の場所等）や手入れの不備による故障
カ　接続している他の機器に起因して故障が生じた場合
キ　その他使用者側の責に帰する故障

また、不良要因が再現されなかった場合につきましては、そのまま製品を返却させて頂く場合もありますのでご了承下さい。

修理は宅配便によるセンドバックで行います。尚運賃は互いに発送する側が負担するものとします。
出張修理は行っておりません。簡単な故障であれば一週間程度で修理可能ですが、故障状況によっては更に日数を要する事もあります。
また、有償修理の場合の修理料金については、簡単な故障で基本料金￥１０，０００円＋部品代とお考え下さい（事務手数料を含む）。修理費用限度額がある場合は、お申し出頂ければ超過する事が明確になった時点で御連絡いたします。

修理品送付先

〒301-0853
茨城県龍ヶ崎市松ヶ丘３丁目１８番地３
フレックスコア
品質管理部　林
E-mail: support@flexcore.jp
Fax:050-3488-3354

６－３　アナログ出力範囲の再調整

１－６項に示すような形で動作確認を行い、出力範囲やオフセットの変化が認められるときは再調整が必要です。アナログ回路は経年環境変化に対する保守を定期的に行うことが望ましく、夏冬の使用環境・周囲温度に差がある場合は季節単位、通年安定した使用環境の場合は１～２年に１度は校正することが理想的です。

再調整の方法・手順を以下に記しますが、被調整対象のＦＣＤＡ１６１６ＰＣＩの他に、高精度の電圧計、各出力を個別に切り替える事ができるスイッチボックス及びWindow7 等がインストールされたパソコンが必要です。

６－３－１　機器間の接続

まず本ボード上のジャンパー（ＳＷ－ＲＭ）の設定を確認します。（Ａ側にプラグ／Ｂ側にプラグ何れか）その後、本ボード上のチェックピン（ＴＰ＋及びＴＰＧ）に電圧計を接続しパソコン及び電圧計の電源を投入します。

６－３－２　暖機運転

一般的に言って、これらの測定器は使用前数時間に渡って通電しておかないと、所期の性能を発揮できません。ＦＣＤＡ１６１６ＰＣＩ自体も、それ程ではありませんがある程度のウォーミングアップが必要です。実際に必要なウォーミングアップ時間については御使用頂く各測定器の取扱説明書により適切な時間を設定するようにして下さい。（ＦＣＤＡ１６１６ＰＣＩ自身については一時間程度のウォーミングアップが必要です。）

６－３－３　調整作業の実際

前項で示す暖機運転時間経過後、表６－３－３Ａに従い、チェックピンへの出力電圧をトリマＴＭ１により調整します。

表６－３－３Ａ　基準電圧調整値

ジャンパーＳＷ－ＲＭの設定	Ａ側にプラグ	Ｂ側にプラグ
基準電圧値	１０．２４０Ｖ±０．００１Ｖ	１０．０００Ｖ±０．００１Ｖ

その後、実際の出力電圧をモニターし動作確認を行います。動作確認プログラムを使用される場合は、下記の動作例を参考に操作を行い、出力コネクタ部でその出力電圧を確認します。因みに、下記の例では出力電圧は表６－３－３Ｂに示す値になります。（ＳＷ－ＲＭジャンパーはＡ側にセット）

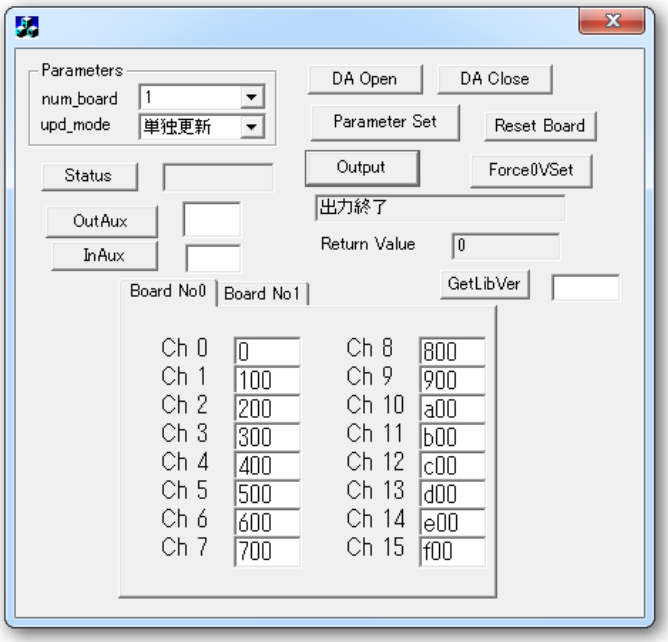


図６－３－３　動作確認プログラム動作例

表６－３－３Ｂ　出力電圧値例

チャンネル番号	出力電圧	チャンネル番号	出力電圧
チャンネル０	（－１０．２４Ｖ）	チャンネル８	０Ｖ
チャンネル１	－８．９６Ｖ	チャンネル９	１．２８Ｖ
チャンネル２	－７．６８Ｖ	チャンネル 10	２．５６Ｖ
チャンネル３	－６．４Ｖ	チャンネル 11	３．８４Ｖ
チャンネル４	－５．１２Ｖ	チャンネル 12	５．１２Ｖ
チャンネル５	－３．８４Ｖ	チャンネル 13	６．４Ｖ
チャンネル６	－２．５６Ｖ	チャンネル 14	７．６８Ｖ
チャンネル７	－１．２８Ｖ	チャンネル 15	８．９６Ｖ

本ボードでは±１０Ｖを超える出力電圧については保証範囲外となります。
上表は、計算値であり、実際の出力はこの値から多少の変動があります。
許容誤差範囲は、夫々中心値に対して±１０ｍＶ以内とします。

第７章　エラー情報

７－１　PCI デバイス指定方法について

インストーラーに対し、PCI デバイスを指定する際、レビジョンコードまでを指定していたため、PCI デバイスのレビジョンが変更されるとインストーラーがインストールを拒否する事実が判明したため、同部分を、デバイスの指定から削除する変更を行った。デバイスドライバーそのものには変更がないため、インストールパッケージのタイムスタンプによる判断とする。（２０１４年４月２８日版から対応バージョンとする。）